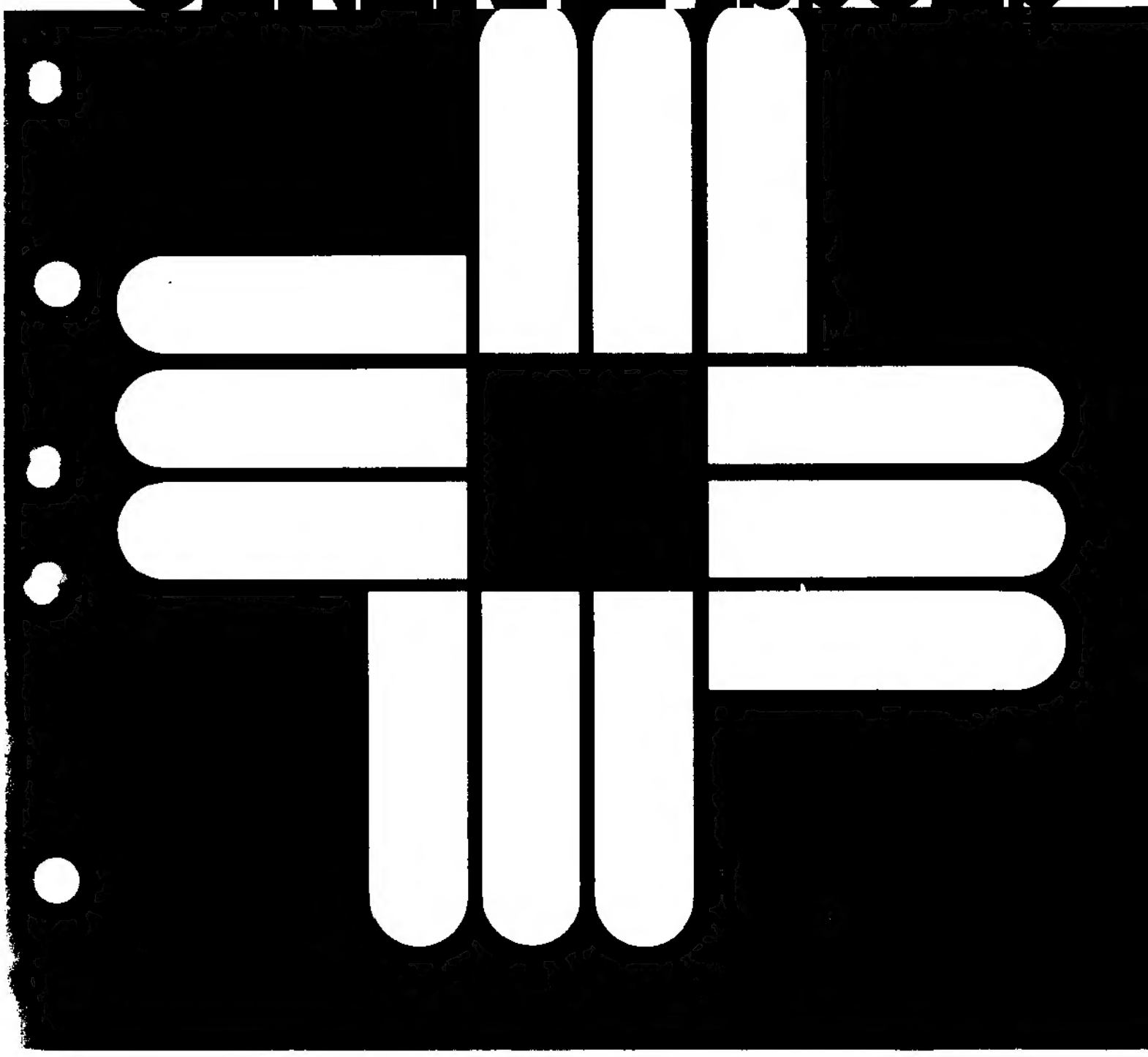
MODCOMP
MODULAR COMPUTER SYSTEMS, INC.

Concepts and Characteristics

MAX IV Operating System

GENERAL ISSUES



MODCOMP
MODULAR COMPUTER SYSTEMS, INC.

Concepts and Characteristics

MAX IV Operating System

GENERAL ISSUES

MANUAL HISTORY

Manual Order Number: 205-804001-I00

Title: MAX IV OPERATING SYSTEM, Concepts and Characteristics Manual

REVISION NUMBER	DATE ISSUED	DESCRIPTION
H00	11/83	Initial Issue (H.0).
I00	02/85	Reissue (I.0). Reflects changes for MAX IV Revision I. Added features include MODCOMP FORTRAN 77, MODACS II, and MAGIC.

Contents subject to change without notice.

PREFACE

Audience: This manual is directed to management, system programmers, system analysts, and sales personnel who need to be familiar with the MAX IV Operating System.

Subject: Summary of the design and functionality of the MAX IV Real-Time Operating System and Support Software product. In addition, optional support software that runs under the MAX IV OS is also discussed.

Products Supported: The CLASSIC II series and CLASSIC 32/85 (in 16-bit mode) computers are supported by this manual. Throughout the text, these products will be referred to, only, as CLASSIC computers.

Related Publications: The reader is referred to the following documents for additional information. When ordering manuals, use the Manual Order Number listed below. The latest revision (REV) will be shipped.

MANUAL ORDER NUMBER	MANUAL TITLE
Operating System Manuals:	
213-804001-REV	MAX IV GENERAL OPERATING SYSTEM, System Guide Manual
213-804002-REV	MAX IV SYSGEN, System Guide Manual
213-804003-REV	MAX IV EXECUTIVE (REX) SERVICES, System Guide Manual
212-804001-REV	MAX IV DATA STRUCTURES, System Design Manual
213-804004-REV	MAX IV OPERATOR COMMUNICATIONS, System Guide Manual
211-804002-REV	MAX IV/MAX 32 NONRESIDENT JOB CONTROL AND BATCH FACILITIES, Programmer's Reference Manual
211-821001-REV	MODCOMP STAND-ALONE SUPPORT SOFTWARE, Programmer's Reference Manual
211-839001-REV	MAX IV MAGIC Programmer's Reference Manual
Basic Input/Output System Manuals:	
213-804005-REV	MAX IV BASIC I/O SYSTEM, System Guide Manual
213-804006-REV	MAX IV UNIT RECORD DEVICE HANDLERS, System Guide Manual

213-804007-REV	MAX IV DATA STORAGE DEVICE HANDLERS, System Guide Manual
213-804008-REV	MAX IV COMMUNICATION HANDLERS, System Guide Manual

File Manager Manuals:

209-804001-REV	MAX IV FILE MANAGER, File Management Manual
211-804019-REV	MAX IV/MAX 32 FILE MANAGER FILE LISTING (FMLIST) UTILITY, Programmer's Reference Manual
211-804018-REV	MAX IV/MAX 32 FILE MANAGER SAVE/RESTORE (FMSAVE) UTILITY, Programmer's Reference Manual
215-804006-REV	MAX IV FORTRAN INTERFACE TO FILE MANAGER, Library Reference Manual

System Processor Manuals:

Program Development:

211-804005-REV	MAX III/IV SOURCE EDITOR, Programmer's Reference Manual
211-804021-REV	MAX IV MODCOMP SCREEN EDITOR, Programmer's Reference Manual
211-804004-REV	MAX IV UTILITY PROCEDURES, Programmer's Reference Manual
299-000000-REV	MAX IV UTILITY PROCEDURES, Pocket Reference Guides
211-804003-REV	MAX III/IV LINK LOADER, Programmer's Reference Manual
211-804009-REV	MAX III/IV LINK EDITOR, Programmer's Reference Manual
211-804010-REV	MAX IV LINK EDITOR, Programmer's Reference Manual
211-804008-REV	MAX III/IV LIBRARY UPDATE, Programmer's Reference Manual

211-804011-REV	MAX IV TASK/OVERLAY CATALOGER, Programmer's Reference Manual
211-804013-REV	MAX IV DEBUG, Programmer's Reference Manual
211-804020-REV	MAX IV DUMP ANALYZER, Programmer's Reference Manual

File/Data Maintenance:

211-804014-REV	MAX III/IV DIRECT ACCESS MAINTENANCE PROCESSOR, Programmer's Reference Manual
211-804015-REV	MAX III/IV MOVING HEAD DISC PREPARATION, Programmer's Reference Manual
210-600500-REV	MAX II/III/IV SYSTEM PROCESSORS BINARY RECORD DUMP, Programmer's Reference Manual
211-804016-REV	MAX IV BACKUP AND RESTORE PROCESSOR, Programmer's Reference Manual
211-804017-REV	MAX IV ALTERNATE TRACK PROCESSOR, Programmer's Reference Manual

Library Manuals:

215-804002-REV	MAX IV MATH LIBRARY, Library Reference Manual
215-804004-REV	MAX IV UTILITY LIBRARY, Library Reference Manual
215-804005-REV	MAX IV EXECUTIVE FUNCTIONS AND PROCESS I/O, Library Reference Manual
220-610437-REV	MAX IV FORTRAN INTERFACE TO FILE MANAGER, Library Reference Manual
215-804006-REV	MAX IV FORTRAN INTERFACE TO FILE MANAGER, Library Reference Manual
220-609404-REV	MAX IV MAGNETIC TAPE LABELED VOLUME SUPPORT, Library Reference Manual

220-610900-REV	MAX IV FORTRAN INTERFACE TO INTERTASK, Library Reference Manual

Language Manuals:

210-804001-REV	MODCOMP ASSEMBLERS, Language Reference Manual
211-804007-REV	MAX III/IV ASSEMBLER CROSS REFERENCE, Programmer's Reference Manual
210-804002-REV	MAX IV FORTRAN IV, Language Reference Manual

Optional Support Software Manuals:

Redundant Software System:

211-832001-REV	LIFEGUARD, Programmer's Reference Manual
212-832001-REV	LIFEGUARD, System Design Manual

Communications:

207-815001-REV	MODCOMP X.25, Communications Reference Manual
211-818001-REV	MAX IV EMULATOR SYSTEM 2780/3780 TERMINALS, Programmer's Reference Manual
212-818001-REV	MAX IV EMULATOR SYSTEM 2780/3780 TERMINALS, System Design Manual
210-612624-REV	MAX IV 3271 BISYNC/EMULATOR, Programmer's Reference Manual
220-612624-REV	MAX IV 3271 BISYNC/EMULATOR, System Design Manual
211-829003-REV	MODACS III PROCESS I/O SYSTEM, Programmer's Reference Manual
211-836001-REV	MODACS II PROCESS I/O SUBSYSTEM, Programmer's Reference Manual

205-805001-REV	MAXNET III AND MAXNET IV, Concepts and Characteristics Manual
211-805001-REV	MAXNET IV, Programmer's Reference Manual
213-805001-REV	MAXNET IV, System Guide Manual
214-805001-REV	MAXNET IV, System Installation Guide Manual
212-805001-REV	MAXNET IV, System Design Manual

File/Data Maintenance:

211-806001-REV	MAX IV PRINTER SPOOLER/DESPOOLER SUBSYSTEM, Programmer's Reference Manual
208-812001-REV	INFINITY DATA BASE MANAGEMENT SYSTEM, Data Management Manual
208-812002-REV	QUERY INTERACTIVE INQUIRY AND REPORT WRITER, Reference Manual
208-812003-REV	QUERY INTERACTIVE INQUIRY AND REPORT WRITER, Technical Manual
211-809001-REV	TSX TIME SHARING EXECUTIVE AND TRANSACTION PROCESSOR, Programmer's Reference Manual

Languages:

210-837001-REV	MODCOMP FORTRAN 77, Language Reference Manual
211-837001-REV	MODCOMP FORTRAN 77, Programmer's Reference Manual

210-813001-REV	PASCAL, Language Reference Manual
210-813002-REV	PASCAL USER'S GUIDE, Language Reference Manual
210-833001-REV	MAX IV COBOL 74, Language Reference Manual
211-833001-REV	MAX IV COBOL 74, Programmer's Reference Manual
211-824001-REV	MAX IV SORT/MERGE SYSTEM, Programmer's Reference Manual
210-608251-REV	CORAL 66, Language Reference Manual

General Issue:

299-100003-REV	DICTIONARY OF TECHNICAL TERMS General Issue Manual

MODCOMP Product Training: The Training Department can supply you with course schedules and contents for training courses provided at our training facility in Fort Lauderdale, Florida. Special courses and alternate locations can be arranged with the Training Department.

MAX IV OS REVISION I.0 SUMMARY

Revision I.0 of the MAX IV Operating System supports three new features:

- MODACS II Process Input/Output Subsystem
- MODCOMP FORTRAN 77 Language
- MAGIC - Multi-User Applications and Guide for Inquisitive Customers

The MAX IV Operating System no longer supports the optional support software:

- BIOS Interface to X.25
- Down Line Load
- UT200 Emulator

MAX IV OS also no longer supports WORKBENCH, the optional development environment, and the X-Y Plotter, optional library support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
OVERVIEW OF THE MAX IV REAL-TIME OPERATING SYSTEM

This chapter contains an overview of the MAX IV Real-Time Operating System (MAX IV OS) and Support Software product, including a definition of the MAX IV OS and introductions to the operating system and to the standard support software. In addition, the optional support software available with the MAX IV OS is defined and the publications set for the system is introduced.

1.1 A DEFINITION OF THE MAX IV OS AND SUPPORT SOFTWARE PRODUCT

The MAX IV Real-Time Operating System (MAX IV OS) is a real-time, communication-oriented, multiprogramming/multitasking operating system. The MAX IV OS specifically supports the advanced hardware features of the medium-to-large scale CLASSIC computers. The MAX IV OS includes I/O device support for all standard peripheral devices that can be attached to CLASSIC computers.

A complete set of program development and file maintenance utility processors is provided. In addition, ASSEMBLER and FORTRAN language support is contained in the MAX IV OS package. Several other languages are also optionally available.

Through the system generation process, the MAX IV OS can be configured in a variety of forms to support an wide range of real-time and dedicated user applications. The MAX IV OS responds rapidly to a large quantity of asynchronous interrupts allowing fast task context switching in real-time environments.

1.2 THE FEATURES OF THE MAX IV OS

The MAX IV Operating System features include:

- Real-Time Multiprogramming/Multitasking - the apparent concurrent execution of two or more programs on a single computer.
- Event-Driven Priority Schedule - the Taskmaster, a resident system element, transfers CPU control according to user-defined task priorities.
- Influence Levels - an inter-task control scheme by which tasks can affect other tasks.
- Interrupt routines - sixteen possible hardware interrupt levels activate executive programs that monitor the execution of system routines and user programs.
- Task Scheduling - the use of timers and external interrupts to activate, resume, or kill a task.
- Allocation of System Resources - memory is dynamically allocated to each task as it is loaded. When the task exits, memory is automatically de-allocated.
- Task Rolling - re-allocation of memory to higher priority tasks.

- o Complete Protection of System Integrity - access rights determine the degree of access for each page of virtual memory.

- o Memory Error Logging - an error logging service that logs the time, type, and location of all parity errors.

- o The SHADOW - a trace/monitor debugging aid.

1.3 THE PARTS OF THE MAX IV OS/SUPPORT SOFTWARE

The MAX IV Real-Time Operating System and Support Software product is composed of:

- o The Operating System Fundamentals
- o The Basic Input/Output System (BIOS)
- o The File Manager (FM) Subsystem
- o System Processors
- o Libraries
- o Language Support

There is also a wide array of optional support software available.

Refer to Figure 1-1 for an illustration of the parts of the MAX IV OS and Support Software.

1.4 THE OPERATING SYSTEM FUNDAMENTALS

The Operating System fundamentals consist of:

- o Data Structures - all tables, lists, and queues used by the MAX IV OS.

- o Request Executive (REX) Services - subroutines that are available to all tasks. These services accomplish the following functions:

  - File Management Services
  - I/O Operation Queuing Services
  - Task Execution Control Services
  - Task Scheduling Services
  - Byte-String Syntax Analysis Services
  - Code Conversion Services
  - Information Services
  - Utility Services

- o Terminal Monitor Program (TMP) - a privileged task that provides a set of features used to control terminals and programs.

- o Operator Communications (OC) Task - provides on-line control of MAX IV operations. PRIORITET STRAX UNDER X-TASK, ENDAST 1 AVVISNODE ÅT GÅNGEN

- o Job Control (JC) Task - facilities for both multiple batch streams and multiple interactive terminals for software development.

- Taskmaster - a resident system element that transfers CPU control according to user-defined task priorities.

- Module Loader (ML) - a quick-load format allows the Module Loader of the MAX IV Operating System to map and load a task or overlay near the speed of the bulk storage device (moving head disc).

- Output Spooling (S) Task - a technique that allows one or more printing devices to serve the needs of many concurrent real-time and batch-processing tasks.

- Exceptional Condition (X) Task - a resident system task that operates at the highest task priority level (0). Its purpose is to report exceptional system conditions to the system operator and to perform task-level operations for the MAX IV Operating System itself.

- Inter-Task Communication - a MAX IV message service that allows for variable length collections of data to be transmitted between tasks.

Refer to Chapter 2 for a detailed discussion of each OS fundamental.

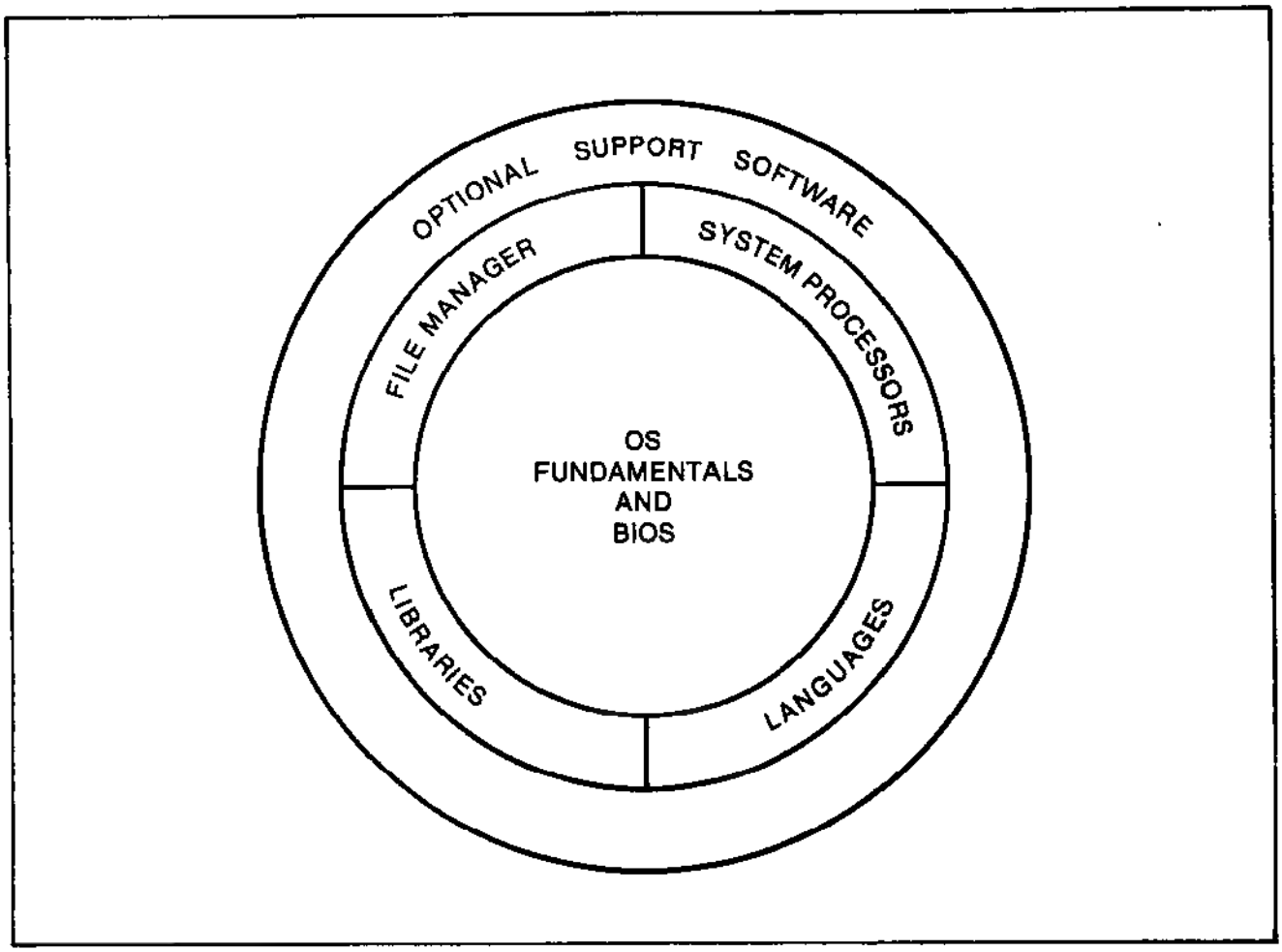


Figure 1-1. Parts of the MAX IV Operating System Support Software

1.5 BASIC INPUT/OUTPUT SYSTEM (BIOS)

The Basic Input/Output System (BIOS) is a common central routine that handles all requests for I/O operations. BIOS schedules and monitors all I/O transactions to the peripheral controllers (or symbiont tasks). BIOS is re-entrant. BIOS may be interrupted at any point by a higher priority process and, once this process is complete, BIOS is resumed at the point of interruption. In other words, the BIOS REX service of the lower priority process does not have to finish before the higher priority process can be serviced.

BIOS is a logical input/output system. That is, a program does not directly access an I/O device. All I/O operations are directed to previously assigned logical files.

Refer to Chapter 3 for a detailed overview of BIOS. Refer to Figure 1-2 for an illustration of the parts of the MAX IV BIOS.

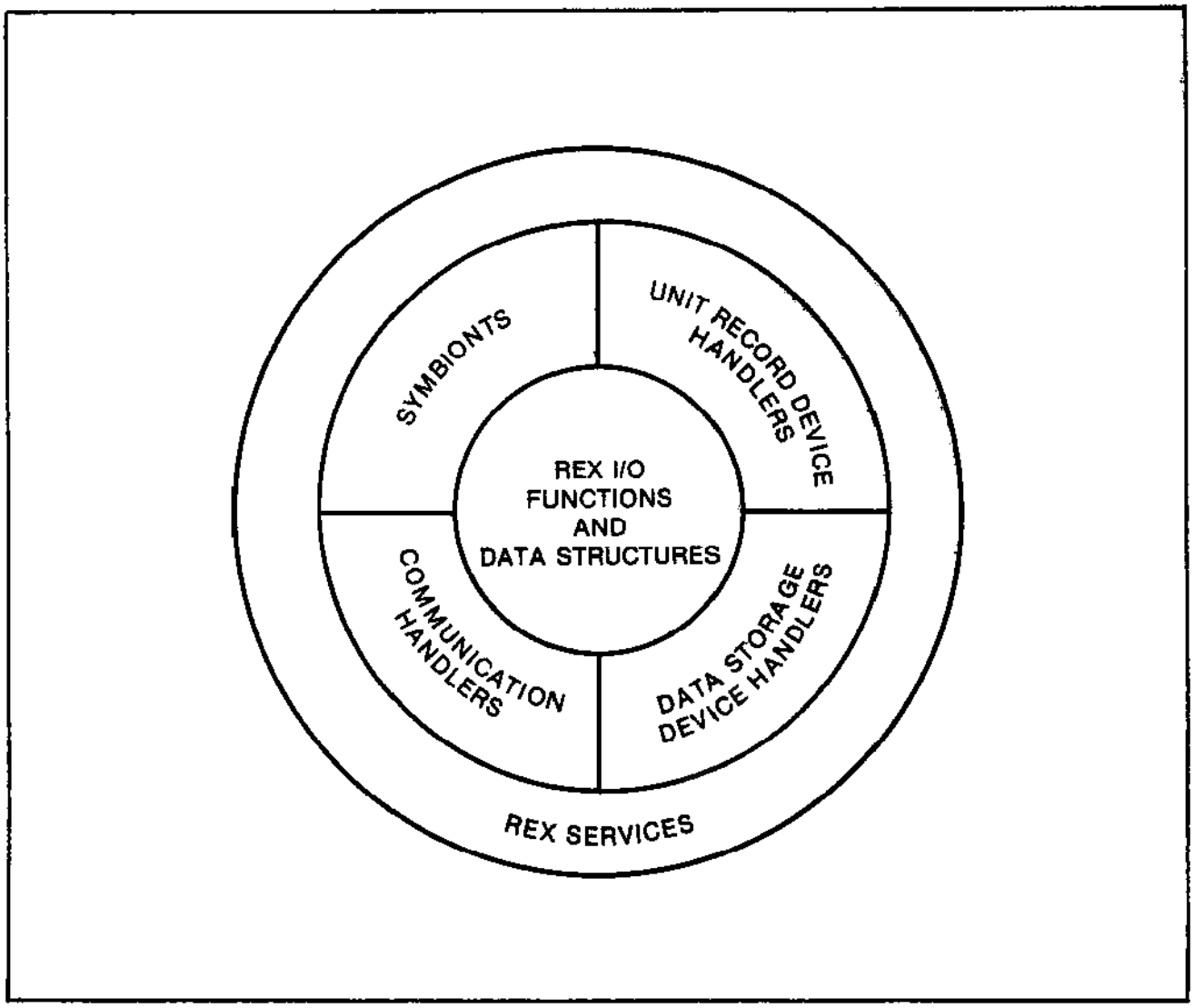


Figure 1-2. The MAX IV Basic Input/Output System (BIOS)

1.6 THE MAX IV FILE MANAGER (FM) SUBSYSTEM (ANVAND. C3 : LEV. SYST) ATS

The MAX IV File Manager (FM) Subsystem is a SYSGENed option of the MAX IV OS that monitors and controls disc usage. The File Manager organizes, maintains, and services multi-level file structures. Data files and directory files can be nested with facilities for volume and file security. Refer to Chapter 4 for a detailed overview of the MAX IV File Manager Subsystem. Refer to Figure 1-3 for an illustration of the parts of the File Manager Subsystem.

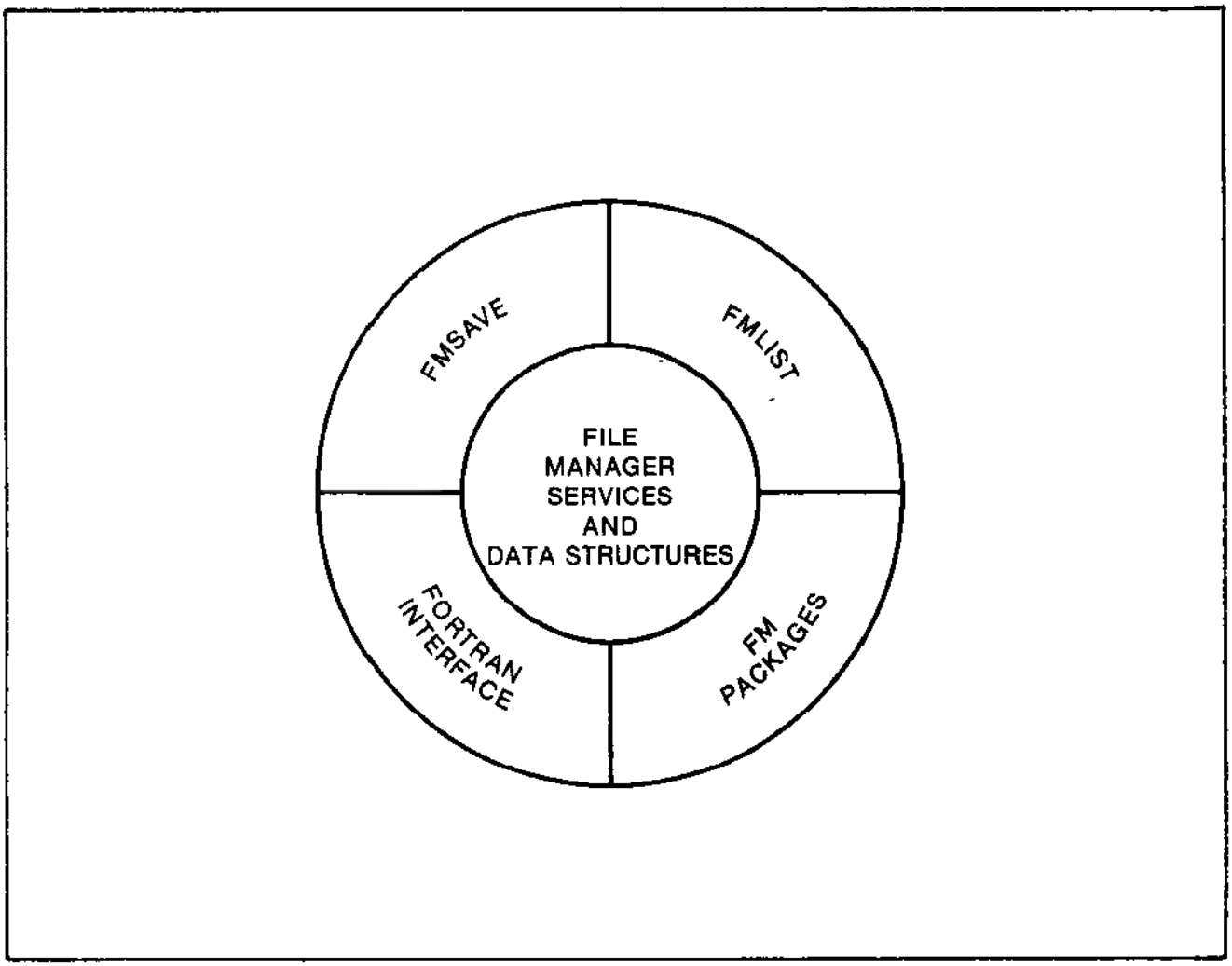


Figure 1-3. MAX IV File Manager Subsystem

1.7 SYSTEM PROCESSORS

The MAX IV System Processors are a collection of system utilities for program development and file/data maintenance. The system processors are the tools with which applications are built and utility functions performed. Refer to Figure 1-4 for an illustration of the system processors and standard languages.

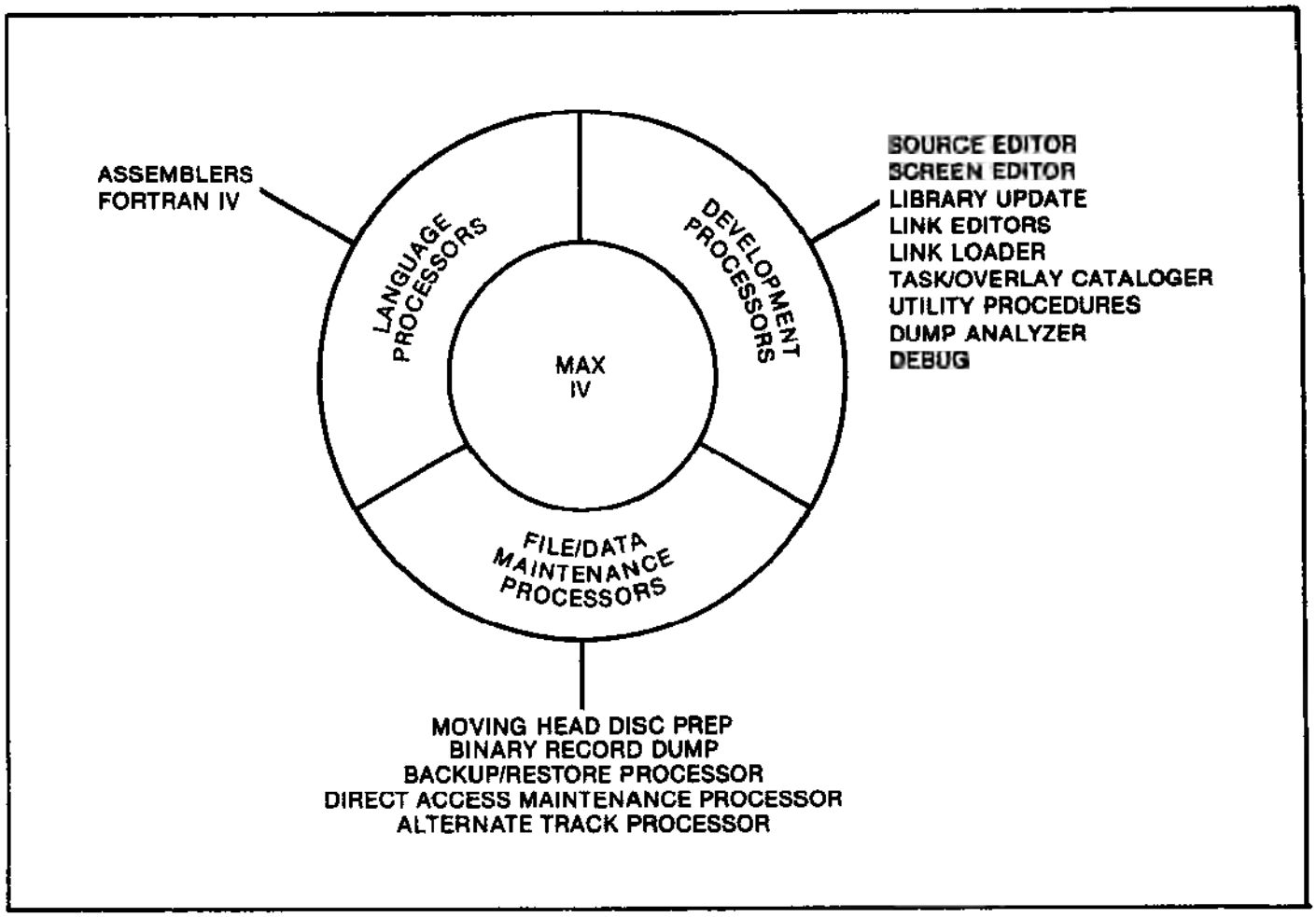


Figure 1-4. System Processors and Languages

The following are system processors used for program development:

- Source Editor (SED) - a sophisticated line editor for source program preparation and maintenance. SED supports both sequential and directory files.
- MODCOMP Screen Editor (MSED) - a display-oriented interactive text editor for source program entry and maintenance. MSED is compatible with SED.
- Utility Procedures - a set of Job Control procedures used for program development and system initialization. These procedures present a standard interface to the MAX IV OS.
- Library Update (LIB) - a system processor used to build and maintain object libraries. LIB supports both sequential and directory files.
- MAX III/IV Link Editor - a link editor used during the Link-Edit Phase of the system generation process to produce output compatible with the Stand-Alone Linking Loader (SAL).
- MAX IV Link Editor - a full-service link editor.

- Link Loader - a system processor that loads a main object module into memory, together with any additional external object modules, and links the modules into an executable set of machine code.

- Task/Overlay Cataloger - a system processor that places fully-edited programs on disc in a directory library for subsequent action by the system loader. This process is known as cataloging. LADO4S SEQAN MED ML (MODUL LOADER)

- Debug - a system processor used to debug machine language programs.

- Dump Analyzer - a set of software provided with the MAX IV OS that allows the entire contents of actual memory to be dumped to either disc or magnetic tape and facilitates analysis of the dump. (AWL4) -(C01)

The following are system processors used for file/data maintenance:

- Direct Access Maintenance Processor (DAMP) - provides the user with services in manipulating a FORTRAN direct-access directory.

- Moving Head Disc Preparation - initializes disc packs.

- Binary Record Dump - displays records written in MODCOMP Standard Binary Mode.

- Backup and Restore Processor - copies data from disc to magnetic tape or to disc, restores data from magnetic tape to disc, and copies data from magnetic tape to magnetic tape.

- Alternate Track Processor - supports the hardware implementation of alternate tracking.

Refer to Chapter 5 of this manual for overviews of each system processor. For detailed reference information on the use of each system processor, refer to the appropriate programmer's reference manual. The manual order numbers and titles of all system processor manuals are listed in the Preface of this manual.

1.8 MAX IV LIBRARIES

The MAX IV OS and Support Software product contains the following libraries:

- Math Library - a collection of mathematical assembly language subroutines callable from FORTRAN.

- FORTRAN IV Run-Time Library - a collection of FORTRAN callable subroutines that provides formatted and unformatted, sequential or direct access input/output facilities, and various file positioning and marking functions.

- Utility Library - a collection of FORTRAN callable subroutines that utilize any function offered by the REX Services, for example, ACTIVATE, KILL, and RESUME.

- Executive Functions and Process I/O - a collection of FORTRAN callable subroutines used in a Process Input/Output environment.

- FORTRAN Interface to File Manager - a collection of subroutines that provide FORTRAN access to File Manager services.

- Magnetic Tape Labeled Volume Support - supports the creation, modification, and reference of magnetic tape volume data structures.

- FORTRAN IV Interface to the Inter-Task Communication (ITC) Service - a collection of FORTRAN callable subroutines that provide a means of using ITC services.

Refer to Figure 1-5 for an illustration of the MAX IV Libraries and to Chapter 6 of this manual for a detailed overview of each library.

1.9 MAX IV LANGUAGE SUPPORT

As part of the standard package, the MAX IV OS and Support Software contains the following languages:

- Assembly (MODCOMP CLASSIC and CLASSIC II family)
- FORTRAN IV (FORTRAN 66)

Refer to Chapter 7 for a detailed overview of these languages.

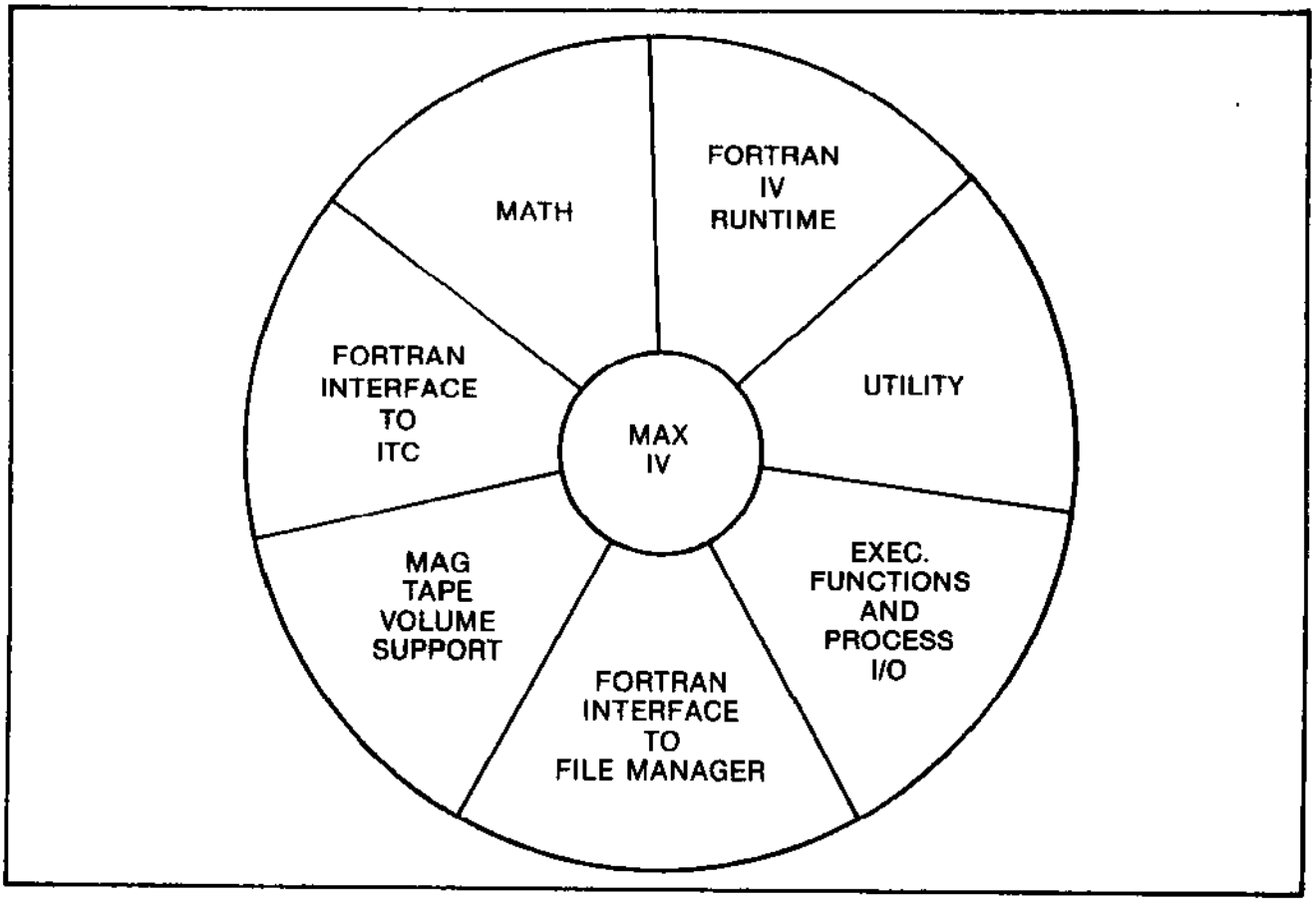


Figure 1-5. MAX IV Libraries

1.10 SYSTEM GENERATION PROCESS

The System Generation (SYSGEN) process is the writing/assembly of several specification statements in a higher-level MACRO language. These statements permit the user to customize the MAX IV Operating System by including only those elements that are required for the user's applications. Refer to Chapter 2, Section 2.3 for a detailed overview of SYSGEN.

1.11 STAND-ALONE SUPPORT SOFTWARE

The Stand-Alone Support Software is a collection of programs for various booting, copying, dumping, and initializing functions. These programs run in a stand-alone mode, that is, without the operating system. Refer to Chapter 2, Section 2.4 for a detailed overview of the Stand-Alone Support Software.

1.12 MAGIC

The Multi-User Applications and Guide for Inquisitive Customers (MAGIC) is a package of software products designed for the multi-user environment. MAGIC includes the following tools and programs to aid in building a unique multi-batch system:

- Electronic Mail
- Electronic Bulletin Board
- Source Compare
- Text Formatter
- Online HELP
- Sign-On Task
- Pre-Scheduled Task Program
- MAGIC Manual

Refer to Figure 1-6 for an illustration of these features and Section 2.5 for a general description.

1.13 OPTIONAL SUPPORT SOFTWARE

In addition to the standard parts of the MAX IV OS and Support Software described in the previous sections, the following optional support software is also available:

- Optional Redundant Software System - LIFEGUARD supports systems that use a back-up central processor.

- Optional communications/networking software:

  - X.25 - the MODCOMP version of the CCITT X.25 DTE/DCE standard for public packet-switched networks.

  - 2780/3780 Emulator - provides emulation of the *IBM (R) 2780/3780 terminal.

  - 3271 Driver/Emulator - front-ends up to thirty-two 3271 Control Units (CU) or communicates with an IBM host while emulating a 3271 CU.

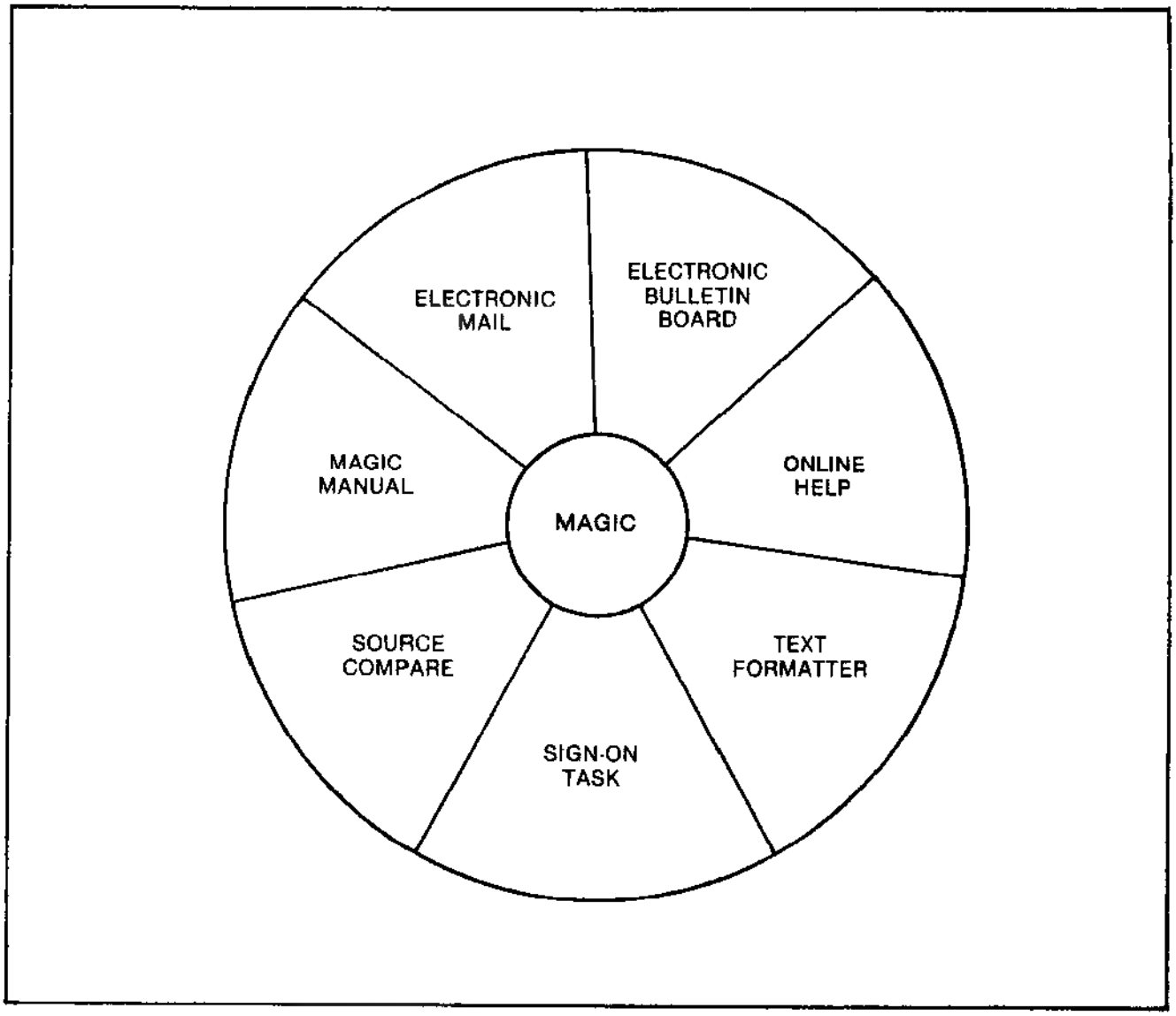


Figure 1-6. MAGIC

- MODACS II Process I/O Subsystem - provides support for the MODACS II hardware system and allows existing application packages to communicate with MODACS II as if it were a local MODACS III hardware system, through minor modifications.

- MODACS III Process I/O Subsystem - provides the software for communicating with the MODACS III hardware subsystems for a Process Input/Output environment.

- MAXNET IV - provides distributed processing capabilities as an extension of the MAX IV OS.

***NOTE:** IBM is a registered trademark of International Business Machines, Inc.

- Optional File/Data Maintenance and Manipulation Software:

  - Printer Spooler/Despooler Subsystem - controls the printing of multiple listings in a multi-user environment.

  - INFINITY - a multi-use data base management system that supports access to data base files through the MAX IV OS logical I/O structure.

  - QUERY - the Interactive Inquiry and Report Writer, a data base inquiry program for INFINITY data bases.

  - TSX - a general purpose Time-Sharing Executive and Transaction Processor.

- Optional language support:

  - MODCOMP FORTRAN 77
  - PASCAL
  - COBOL 74 with SORT/MERGE
  - CORAL 66

Refer to Chapter 8 for a detailed overview of optional support software. Refer to Figure 1-7 for an illustration of the optional support software.

1.14 PUBLICATIONS SET

An integral part of any software package is the publications set that accompanies the software. The MAX IV OS has an extensive and detailed documentation tree. Refer to Chapter 9 for a complete discussion of the MAX IV publications set.

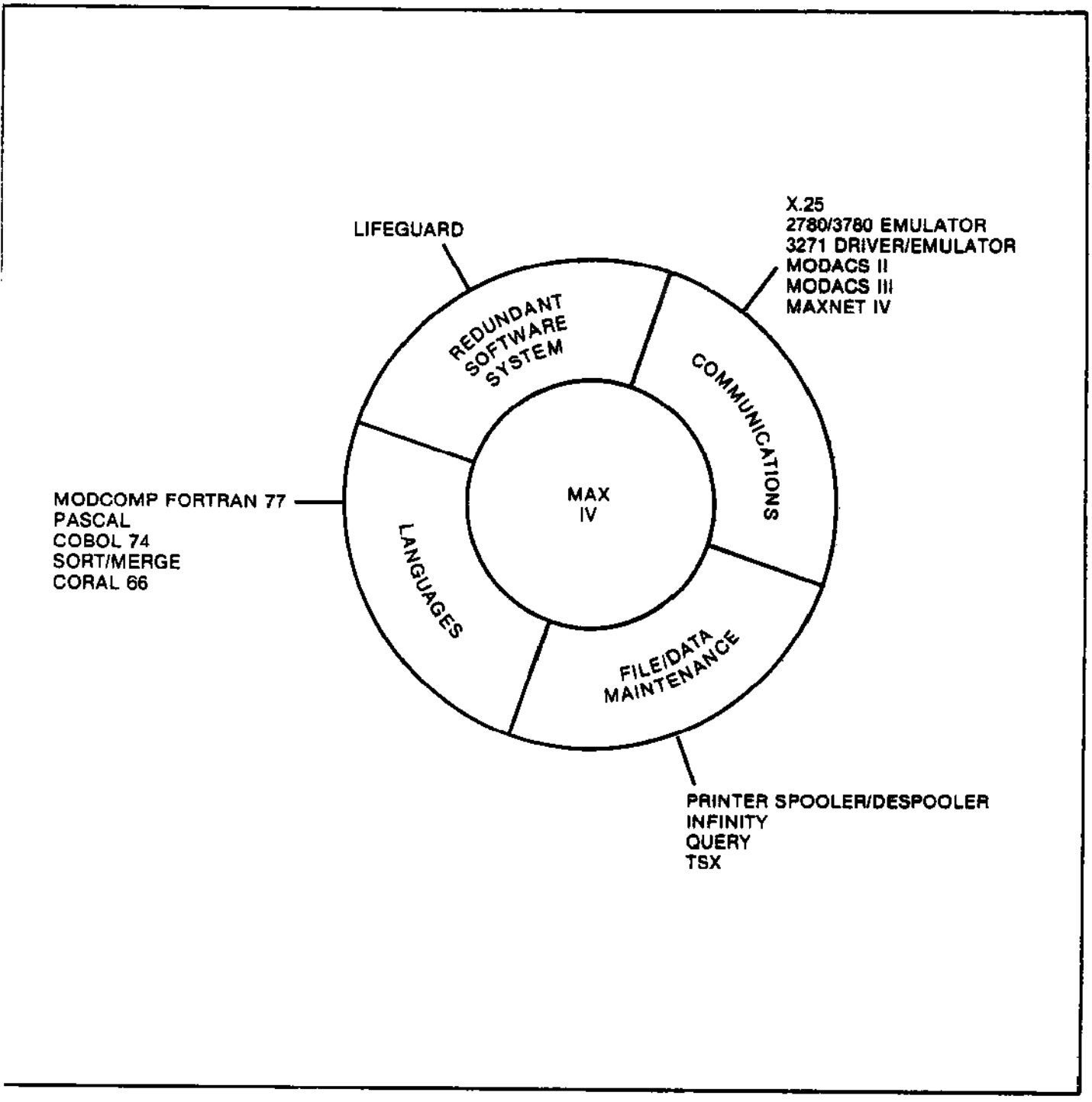


Figure 1-7. Optional Support Software

CHAPTER 2
THE OPERATING SYSTEM

This chapter provides a discussion of each feature and fundamental part of the MAX IV Operating System. In addition, the system generation process and the Stand-Alone Support Software are discussed in this chapter.

2.1 OPERATING SYSTEM FEATURES

The features of the MAX IV Operating System are:

- o Real-Time Multiprogramming/Multitasking
- o Event-Driven, Pre-emptive Priority Schedule
- o Influence Levels
- o Interrupt routines
- o Task Scheduling
- o Allocation of System Resources
- o Task Rolling
- o Complete Protection of System Integrity
- o MAX III Compatibility
- o Memory Error Logging
- o The SHADOW

For detailed information, refer to the MAX IV Operating System, System Guide.

2.1.1 REAL-TIME MULTIPROGRAMMING/MULTITASKING

Multiprogramming/multitasking is the apparent concurrent execution of two or more programs on a single computer. Programs rarely have continuous use of the CPU for more than short periods of time. Most programs become wait-bound quickly, usually while waiting for I/O operations to complete, and leave the CPU in an idle state. A computer can only work efficiently when several programs are available in main memory. When one program becomes idle, there is a good chance that another can utilize the time and computing power that would otherwise be wasted. The term multiprogramming is used to describe this concurrent (but non-simultaneous) program execution on a single CPU (programs run independently). The term multitasking describes the same process, except programs can interact.

The basic unit of multiprogramming/multitasking is the task. A task is a complete program unit that is sufficiently described to permit it to execute in a manner that is concurrent with and independent of other program elements. An interrupt routine, usually called a directly-connected interrupt routine, is invoked automatically by the hardware when an exceptional condition is detected. The term task is applied to those system and application programs that execute under the control of the MAX IV Operating System -- at software-generated interrupt levels called task priority levels.

2.1.2 EVENT-DRIVEN, PRE-EMPTIVE PRIORITY SCHEDULE

Task programs can be assigned to 256 unique software priority levels. More than one task can execute at the same priority level, which forces them to share time on a cyclic basis (if a system ROUNDROBIN feature is optioned). The maximum number of tasks that can exist in the operating system and be active at any one time is usually limited by main memory and secondary storage resources rather than the number of priority levels.

The highest priority task levels are designed specifically for the execution of important real-time tasks. These levels are sometimes called "foreground" levels. Level 0 is the highest level, though it is represented by the lowest number in absolute value.

The lower priority levels are designed specifically for the execution of less important real-time tasks and batch-processing tasks. These levels are sometimes called "background" levels. Level 255 is the lowest level, though it is represented by the highest number in absolute value.

In MAX IV there is no memory boundary that determines whether a particular task is "foreground" or "background". Normally, tasks with priority 0 through 127 are referred to as foreground tasks and those with priority 128 through 255 as background.

A system programmer need only install un-debugged, unprivileged tasks at priority levels below any critical tasks in order to ensure system integrity. Memory space will always be obtained by high priority tasks if enough space is configured in the system and if needed memory space in use belongs to lower priority rollable tasks. Nonresident programs, which require contiguous memory, only need contiguous virtual addressing space in MAX IV. Memory maps permit MAX IV to construct contiguous virtual addressing spaces from noncontiguous actual memory pages.

2.1.3 INFLUENCE LEVELS

The execution priority level, assigned to a task by the system programmer, only determines the "winner" when a conflict exists between two or more tasks that are attempting to use a "basic" system resource such as:

- CPU
- Memory
- Register Blocks
- Memory Maps
- I/O Devices

Unless further restrictions are imposed, any active task is still capable of affecting another active or inactive task, regardless of the individual execution priorities. If both tasks are user tasks, no real destruction can occur because the rules of privileged instructions and the natural isolation of hardware mapping prevents it. However, one task can activate, resume, kill, schedule, or modify the communication variables of another -- which indicates its capability to affect another task.

MAX IV has a separate inter-task control and cooperation scheme that can be invoked by a system programmer when the task is cataloged or when a system generation is performed. In the scheme, each task has a characteristic called "influence" that is similar to, but distinct from, priority. The absolute value of the influence number (a number between 0 and 255) determines a task's relative ability to perform a system function that affects another task. A task is allowed to affect another task only if they are compatible. Compatibility exists when the affecting task has a lower or equal **number** (absolute value) than the affected task. For example, a task with an influence level number of 20 can only affect tasks within the influence level range of 20 through 255. However, since the influence level is inversely proportional to the level number, the compatibility criteria can be restated as: tasks are compatible when the affecting task has a higher or equal influence **level** than the affected task.

To summarize:

0 (highest influence level; lowest number)
.
.
.
255 (lowest influence level; highest number)

The only relationship between task priority and task influence occurs when the CHANGE priority service is executed. A task can be changed to a priority level no higher than the influence level of the task that requests the change. (For example, a task at influence level 20 can change another task's influence level to values in the range 20-255.) Also, priority and influence share the same inverse number range (0 to 255).

2.1.4 INTERRUPT ROUTINES

MAX IV is an interrupt-driven operating system. Sixteen possible hardware interrupt levels activate interrupt routines that drive the execution of system routines and user programs. In addition, any interrupt level can be triggered from software.

The Taskmaster routine, which is triggered by the lowest interrupt level (#F), determines the next task to be run. If a task requires hardware resources, the Taskmaster either assigns them from available pools or causes a reallocation from the lowest priority tasks awaiting execution.

Some active MAX IV components are directly connected to hardware interrupt levels permanently. These components are called standard interrupt programs. In most cases, these interrupt levels are enabled except during critical instruction sequences. All interrupt routines are permanently resident in main memory.

A user may add custom directly-connected components to unused interrupt levels (#7 through #B) if these programs must meet critical response time requirements, and can afford to operate above the task levels.

The following standard directly-connected interrupt programs are included in MAX IV:

LEVEL	NAME	PURPOSE
#0	Power Failure/Auto Restart Interrupt	Provides for orderly shutdown and restart of MAX IV during power cycling.
#1	Hardware Fault Trap	Detects, analyzes, and corrects memory parity errors.
#2	Program Exception Trap	Processes violations of memory access rights and privileged instructions.
#3	Multiprocessor Interrupt	Used for communications between other CPU's (optional); also used as a debug trace/monitor interrupt with the SHADOW.
#4	Unimplemented Instruction Trap	Processes entry of all Executive (REX) Service calls and optional instruction simulations. Provides an orderly transfer between unprivileged and privileged execution states.

#5	Arithmetic Exception Trap	Processes floating-point overflow/underflow violations and normalizes the results.
#6	Clock Interrupt	Updates critical timers directly and triggers a lower priority level (low clock) to process other less-critical system timers. Timers updated at this level include CPU Utilization Timer (current task only) and System CPU Utilization. The CPU Utilization bar graph and the memory viewer are also updated.
#7-#B	External levels	Optionally used by MAX IV as task schedulers.
#C	I/O Data Interrupts (64 sublevels)	Processes data characters and words from slow I/O devices or processes End-of-Block signals from high-speed DMP devices. Each device controller channel has a unique interrupt routine entry, but similar channels can share a common handler.
#D	I/O Service Interrupts (64 sublevels)	Processes initiation and termination of each I/O operation and maintains queues of waiting operations to be performed by each controller channel. Each controller channel has a unique interrupt routine entry, but similar channels can share a common handler.
#E	Console Interrupt and Low Clock	Terminates and restarts the standard Operator Communications task. The level is also used to update the less critical timers deferred by level 6. Such timers include: o Task-scheduling timers o A delay timer for each active task o I/O watchdog timers o The Time-of-Day Clock
#F	Taskmaster Interrupt	Because this level is the lowest priority hardware level, it allocates the CPU time into many task levels. It also allocates some execution resources (a register block and one or two memory maps) to the currently executing task when control is given to it. The Taskmaster context-switches from one task to another when certain events occur that request the level. These events include: o Completion of a delayed or asynchronous service. o Activation of a directly-connected interrupt, such as an expired timer or I/O service completion.

- Return of a system resource to a pool of resources.
- Direct request from another task.

Figure 2-1 illustrates the relationship between the hardware (interrupt) priority levels and the software (task) priority levels created by the Taskmaster interrupt level. The task priority levels are separate from the hardware interrupt levels. All references to hardware levels are hexadecimal notation (#0 to #F) and all references to software task levels use decimal notation (0 to 255).

2.1.5 TASK SCHEDULING

Any task can be scheduled to be activated (started), resumed (continued after pause), or killed (abnormally terminated) by one or more components called Task Schedulers.

One task can schedule another task directly with a REX Service or by using interrupts or timers. The Operator Communications task has standard directives that permit it to schedule another task using a Task Scheduling Timer or Interrupt.

Task Schedulers are of the following types:

- Timers
- External Interrupts

2.1.5.1 Task Scheduling Timers

Timers may be allocated by the CONNECT REX Service from a pool of such timers. The expiration of these timers can be made to schedule a task. Such timers can be set in either of two formats:

- Elapsed time (relative to when set)
- Time-of-day (relative to midnight)

They may be used as one-shot timers, which ensures that they automatically return to the pool once they expire and perform their scheduling function, or as periodic timers, which allows them to expire and perform their scheduling function periodically and continuously until they are manually deallocated.

A single Delay Timer may be used by each task in the system. These timers are similar to the Task Scheduling Timers in the timer pool, except they can only perform the Resume Execution function when they expire and they cannot be made periodic. These timers provide a dedicated time base that the task can set and refer to as needed without affecting the task's execution.

2.1.5.2 Task Scheduling Interrupts

Unused external interrupts may be specified at system generation time for use in task scheduling. A re-entrant interrupt service routine is directly connected to these levels. Such levels may be allocated as task schedulers by the CONNECT REX Service. The scheduled task becomes active, resumed, or killed at its software priority when this interrupt is requested externally by a hardware signal or internally by the SIR command. These schedulers can also be made one-shot (requiring enabling each time) or periodic (requiring enabling only once).

A map zero resident routine can take advantage of the DC Interrupts feature of MAX IV, whereby a module is linked into the interrupt during the SYSGEN process.

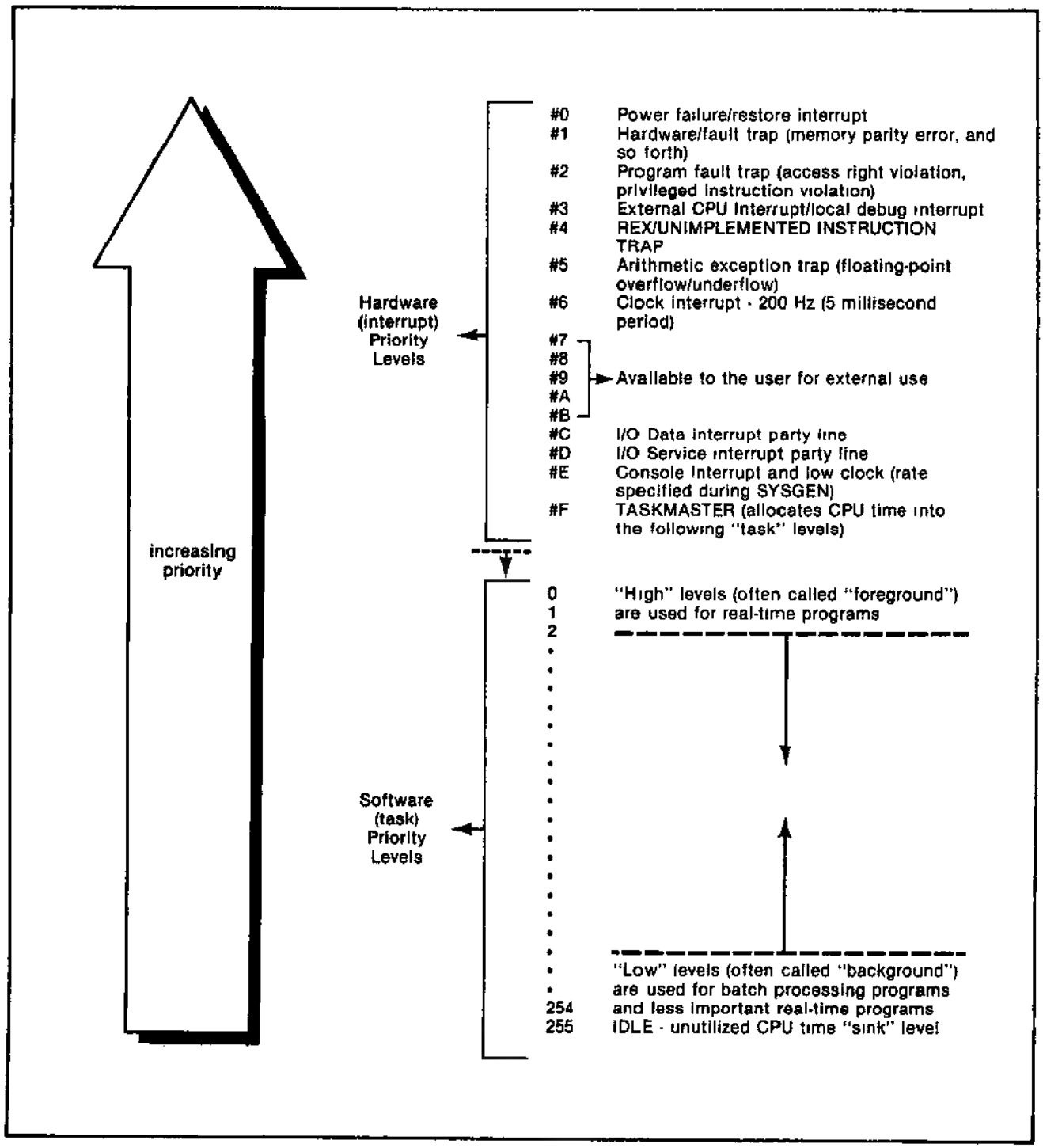


Figure 2-1. MAX IV Hardware/Software Priority Structure

2.1.6 ALLOCATION OF SYSTEM RESOURCES

The memory-resident elements of the MAX IV OS permanently occupy the lowest pages of actual memory. These elements are specified by a system generation process and include:

- Dedicated cells for hardware interrupt vectors, Direct Memory Processor (DMP) channels and MAP 0's map image.
- Request Executive Service (REX) routines. I/O device handlers and interrupt-driven elements.
- Fixed buffers for nonresident services and directives.

The remaining memory forms a pool to be allocated for transient memory requirements during system operation.

2.1.6.1 Main Memory Allocation

Memory is dynamically allocated to each nonresident task as it is loaded. These allocations are made on a multiple page basis and are recorded in the task's map image. Active tasks can request additional memory at run time. When the task exits, memory is automatically de-allocated.

Tasks located in MAP 0 are permanently resident and privileged and are loaded with the MAX IV OS. MAX IV constructs a contiguous virtual addressing space from scattered pages of actual memory. The number of tasks residing in memory simultaneously is limited only by the number of available Task Control Blocks in MAP 0 and actual memory. In all cases, a pointer to a task's map image is maintained within the operating system. The map image is loaded into a hardware map whenever necessary to resume execution.

2.1.6.2 Two-Map Tasks

Since the CLASSIC hardware permits unique map assignments for both instructions and for operands, facilities for executing very large programs (up to 256K bytes of mapped addressing space) are provided. Such a task is assigned to two available memory maps and is called a two-map task (or a split task). Figure 2-2 illustrates how such a large program might appear.

2.1.6.3 Dedicated Resource Items

Pools of resources that are sized at system generation time have individual items or groups of items that may be dedicated to tasks with compatible influence levels for example:

- Timers
- I/O Devices
- Global Shared Regions

Such resources can play the role of "the other task" and can be given influence levels just as tasks. When a task that has been cataloged with a low influence level tries to allocate resource items from such a pool and only items remain with higher influence levels, then the task is given a "pool empty" response even though dedicated resources remain in the pool for those tasks that request them with a sufficiently high influence level.

2.1.6.4 Memory Resources

The allocation of actual memory pages to pages of virtual addressing space is managed by certain Executive Services. Any number of actual memory pages (256 words per page) scattered throughout actual memory may be put together to form any required contiguous virtual space for use by a program.

Because this feature eliminates a common memory fragmentation problem, such techniques as multiple memory pools and fixed memory partitions are unnecessary compromises for MAX IV. A single Available Memory Pool is all that is necessary for MAX IV to service the many asynchronous demands for memory space.

2.1.7 TASK ROLLING

Task rolling is the re-allocation of a lower priority task's memory resources to a higher priority task. The MAX IV OS provides two types of rolling:

- o Automatic
- o Manual

2.1.7.1 Automatic Rolling - The Roller (ROL) Task

The ability of a task to retain its main memory resources is determined by the task's execution priority level. Since batch-processing tasks and real-time tasks under development execute at low priority levels, their main memory resources are subject to be taken away at any time -- if higher priority tasks need these resources. Real-time tasks under development should be executed at priority levels higher than batch-processing levels, as a general rule. This ensures that the real-time tasks are the last tasks to lose their memory when memory becomes scarce and that the batch-processing tasks are the "first-to-go". When any task needs memory and no memory can be found in the available memory pool, then the "memory allocator" queues the request to the Roller task. This task attempts to find the lowest priority task that is designated as being rollable and has enough memory pages to satisfy the demand. This task utilizes a special I/O technique that rolls-out the lowest priority task(s) from memory to disc at the speed of the Direct Memory Processor (DMP) transfer.

The Roller can optionally be made a resident system task or simply be allowed nonresident MAP 0 space. The Roller has the ability to move its effective execution priority level in the CPU queue to a level appropriate to service the highest priority task needing memory. When the available memory pool is capable of serving all task memory requirements, the ROL task moves itself to the bottom of the CPU queue and stays there until some task needs more memory than is available in the available memory pool.

As soon as the number of physical pages required by the rolled task become available, they are mapped into the requesting task's virtual addressing space and the task is rolled-in from disc to memory at the same high speed made possible by the Direct Memory Processor technique. Other tasks or programs of equal priority are rolled in and out of memory, making use of all available resources. When a task is rolled-out, active I/O buffers and associated data structures remain in memory. This allows waiting I/O requests to proceed to completion.

2.1.7.2 Manual Rolling - ROLL REX Service

The ROLL REX service can override the Roller Task. Using this service, a named task can be requested to be rolled out of main memory or to be rolled back into main memory. It can also change the rollability of a task. When a task is requested to be rolled out of main memory, it stays rolled out until a REX ROLL IN service request occurs.

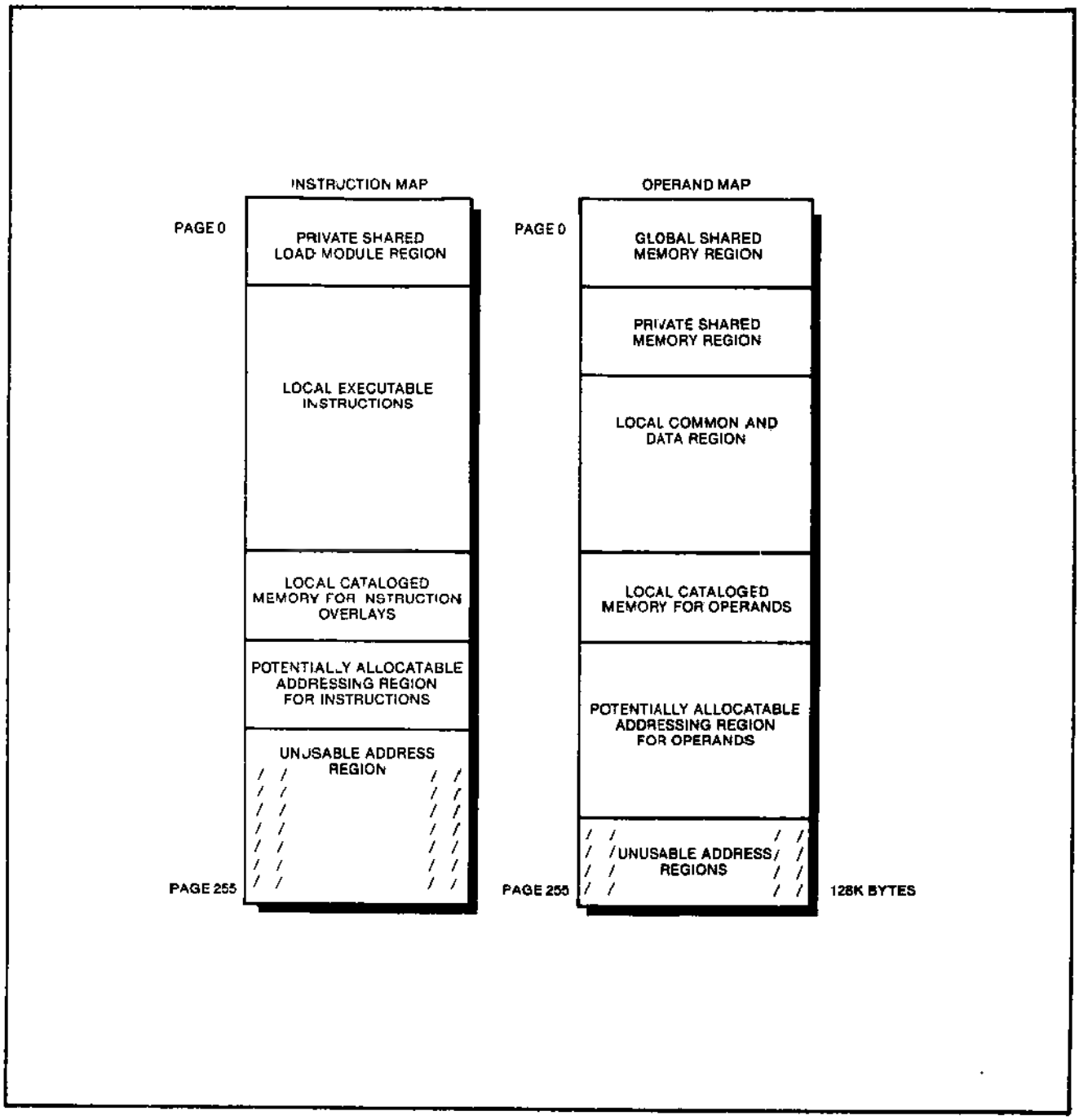


Figure 2-2. The Addressing Space of a Typical Two-Map Task

2.1.8 COMPLETE PROTECTION OF SYSTEM INTEGRITY

Memory protection is provided through the use of memory maps on a 512-byte page basis. Protection is achieved by Access Rights that are associated with a page of virtual memory. These Access Rights determine the degree of access that a particular task is granted to each page. Access Rights control access to memory by all the system elements including I/O. The Access Rights of a virtual page may be one of the following:

Code	Restriction	Use
'00	No access	Unused memory
'01	Read only	Protected data
'10	Read and Execute only	Protected program and constants
'11	Read, Execute, and Write	No restrictions

All I/O and resource manipulation instructions (including halt instructions) are considered privileged and are typically used only by the operating system.

This protection scheme causes a trap if any program violates the Access Rights of any addressing regions assigned to it. The trap may be used to abnormally terminate the program or to return such attempts to a special routine in the program for processing as it chooses.

The advantage of this virtual access right protection over actual memory page protection is that pages of memory shared by several tasks can be given different access rights, which permits some programs to produce data that other tasks can access -- but not modify. The Access Rights may be changed dynamically to permit tasks to communicate through common areas without interaction problems.

Programs that separate their modifiable data areas from non-modifiable instructions and constants are rewarded with valuable debugging aids that catch program failures before the evidence of what caused the failure is lost.

All user tasks are protected from each other as well as being isolated from the system. In addition, major elements of the operating system are protected from other elements of the operating system, which makes debugging new privileged system elements (I/O handlers, REX services, and system tasks) easier and less prone to software failure.

2.1.9 MAX III COMPATIBILITY

During system generation, a user may elect to include the MAX III executive service interfaces. MAX IV can then run most programs written for MAX III without recompilation or re-assembly. The basic calling sequences of the MAX III services have been preserved. However in many cases they have been expanded in a manner that ensures upward compatibility. Certain system processors require that the MAX III services be included.

Task programs and overlays that were written to run under the MAX III Operating System operate normally under MAX IV as long as their only interface to the operating system is by direct REX Service calls or FORTRAN CALL Subroutines. Refer to the MAX IV EXECUTIVE (REX) SERVICES, System Guide Manual, listed in the Preface. If these programs directly access parts of the MAX III Operating System's data structures in main memory, instead of using the service calls provided, then changes may be necessary.

User-coded I/O handlers, symbionts, customer-designed REX services, and other privileged elements designed to operate as part of the MAX III Operating System require changes if they are to be integrated into MAX IV.

The major difference between MAX IV and its predecessors is in the organization of its internal data structures and the methods by which these structures are accessed. These changes are mainly due to the memory mapping structure of the CLASSIC computer. In the older operating systems, such data structures were located in the single fixed 64K address range shared by the operating system and all programs that operated under it.

It was easy for an experienced programmer to access (but not necessarily change) any system data structure directly even though REX Services and MACROS were provided for delivering such information. However, because of the memory mapping structure of the CLASSIC computer, such data structures are not located in the same virtual addressing space (map) with the user's task program and, in some cases, not in memory at all. (Some data structures are maintained in dedicated hardware registers so that the system can process them more efficiently.)

Resident system tasks are located in the same addressing space with the operating system, but the data structures available to them are in a format foreign to a MAX III program element.

The way in which programs are actually loaded into main memory and assigned space in memory is different in MAX IV. The way in which unprivileged programs are prevented from disturbing the operating system and other programs is also different. Such differences are normally transparent to the programmer of the previous operating system and are actually easier to use and understand.

Most operator directives, Job Control statements, and system processor statements are functionally the same although some minor differences exist. The MAX IV Task/Overlay Cataloger and the MAX IV Link Editor are notable exceptions to this rule.

2.1.10 MEMORY ERROR LOGGING

MAX IV provides an error logging service that logs the time, type and location of all parity errors. Services are also provided to attempt recovery from parity errors on those CPU models that do not run with error correction permanently enabled. With MOS memory, the Error Correction Mode of operation will be invoked, and a retry attempted when a memory parity error occurs.

Operator directives are available to enable or disable Error Correcting Mode on a per-plane basis, and to obtain a summary of errors that have occurred.

Enable/disable of successful retry logging is controlled through Operator Control directives and system generation statements. This option will cause all successfully retried I/O operations to be registered with the event logging package.

2.1.11 THE SHADOW

The Shadow is a system debugging tool. The Shadow is an optional interrupt routine operating at interrupt level 3. This routine samples system status at any desired rate, even up to the choke-point of the system. As it samples system status, the Shadow stores the Program Status Register (PS), Program Address Register (PR), and optionally the general registers of the program it interrupts.

Interrupt level 3 is normally used for inter-CPU communications but can be taken over by the Shadow. This interrupt level must be connected to an external pulse generator. The pulse repetition rate must be set according to the function being performed by the Shadow.

2.2 OPERATING SYSTEM FUNDAMENTALS

The Operating System fundamentals consist of:

- Data Structures
- REX Services
- Terminal Monitor Program (TMP)
- Operator Communications (OC) Task
- Job Control
- Taskmaster
- Module Loader (ML)
- Output Spooling (S) Task
- Exceptional Condition (X) Task
- Inter-Task Communication

2.2.1 DATA STRUCTURES

The term data structures is used to refer to all tables, lists, and queues in the MAX IV OS. All these structures are addressable when a program or service accesses the MAP 0 virtual addressing space.

Data structures are organized into six groups:

- Dedicated Memory Cells
- Dedicated Register Blocks
- General System Data Structures
- Basic I/O System Data Structures
- Task-Related Data Structures
- Specific System Task Data Structures

For detailed information on every word and field in the MAX IV Data Structures, refer to the MAX IV DATA STRUCTURES, System Design Manual, listed in the Preface.

2.2.2 EXECUTIVE (REX) SERVICES

The MAX IV Executive (REX) Services allow the user to execute privileged instructions under the supervision of the operating system. These executive services are referred to as REX Services or REX calls. Any task (root task or overlay) may call REX Services by invoking the Request Executive Service (REX) machine instruction or by using one of the many macros and interface subroutines available in higher level languages such as FORTRAN. A user may code re-entrant and recursive REX services and add them to the system at SYSGEN time.

NOTE: Directly-connected interrupt routines coded by the user may not call these services.

The usage and calling conventions of the MAX IV Executive Services are described in the MAX IV EXECUTIVE (REX) SERVICES, System Guide Manual, listed in the Preface.

2.2.2.1 REX Service Features

REX Services execute privileged instructions and modify critical system data structures. Normally, an unprivileged user-coded program would not be allowed such freedom. However, such a program can call REX Services that execute privileged instructions if such activities are supervised by the operating system.

REX Services are re-entrant subroutines located in the nucleus of the operating system, that is, in the virtual addressing space defined by MAP 0. However, REX Services may be called from any virtual addressing space because the execution of a REX instruction causes a machine trap (hardware interrupt). A trap can branch across map boundaries.

Even though a REX Service subroutine is located in MAP 0 addressing space, it can fetch arguments from the addressing space of the calling task and return such arguments into that addressing space. Special privileged instructions are used to accomplish this movement of data from one addressing space to another. Arguments may also be passed by using the registers of the calling task because these registers are directly accessible by the REX Service subroutine. In fact, the subroutine uses the same block of general registers as the calling task.

An execution of the REX machine instruction activates the Unimplemented Instruction Trap (hardware interrupt level 4). The REX Service interrupt routine is then entered, provided level 4 or higher levels are not active.

Even though the REX machine instruction causes a hardware interrupt, this high priority lasts only until the service is selected. The Executive Service itself operates at the software priority level of the calling task, except while updating or testing certain data structures.

2.2.2.2 Types of REX Services

The following types of REX Services are included in the MAX IV OS:

- Basic Input/Output
- Task Execution Control (self-initiated)
- Execution Control (to other tasks)
- Task Scheduling
- Priority Change
- Task Relinquish
- Job/Task State
- Exclusive Use
- Establish Residency
- Memory Allocation
- Loader
- Trap Control
- Byte String Analysis
- Format/Data Conversion
- Event Logging
- Task Rolling
- Memory Dump
- Time and System Status
- Intertask Communications
- Task Status
- Task Option and Variable Control
- Modification of Memory Access Rights

2.2.2.3 Calling REX Services

Executive services may be invoked in the following ways:

- o Direct call
- o FORTRAN call
- o Operator Communications Directives
- o Job Control Directives
- o Terminal Monitor Program Directives

A. Direct Calls of REX Services

The most direct way to execute a REX instruction is in ASSEMBLY language.

EXAMPLE

```
LDI,R8      REX servicenumber
REX,MAXIV
```

Depending on the service, various calling arguments may be required to be pre-loaded into registers or included in-line following the REX instruction. Arguments returned by the execution of the service may be in the registers when the service returns to the calling program. REX Services do not modify any registers other than those used for passing or returning arguments. The System Equate File (IVEQUATES) provides the ability to call REX Service numbers and options as names. Refer to the MAX IV EXECUTIVE (REX) SERVICES, System Guide.

B. FORTRAN Calls of REX Services

The following format of calling REX Services adopts FORTRAN (FR5) subroutine conventions. The ISA FORTRAN PROCESS I/O calling standard is followed.

EXAMPLE

```
CALL verb ([argument],...)
```

FORTRAN calls exist for all REX Services, even though FORTRAN programs can use direct Assembly language forms by using INLINE/FINI statements. Refer to the MAX IV FORTRAN IV, Language Reference Manual, listed in the Preface.

C. Operator Communications Directives

The system operator can execute many REX Services that exert control over other tasks by entering directives that are interpreted by the Operator Communications Task.

EXAMPLE

```
/taskname/verb [argument]...
```

Refer to the MAX IV OPERATOR COMMUNICATIONS, System Guide Manual, listed in the Preface.

D. Job Control Directives

The batch-processing programmer can execute many REX Services by entering Job Control Directives at the beginning of a job or between job steps.

EXAMPLE

```
$verb [argument]...
```

These services can only affect resources that have been committed to batch-processing tasks. Refer to the MAX IV/MAX 32 NONRESIDENT JOB CONTROL AND BATCH FACILITIES Programmer's Reference Manual, listed in the Preface.

E. Terminal Monitor Program Directives

Executive services can also be invoked through directives in the Terminal Monitor Program (TMP). Refer to Section 2.2.3.

EXAMPLE

```
/verb [argument]
```

2.2.3 TERMINAL MONITOR PROGRAM (TMP)

The Terminal Monitor Program (TMP) provides a set of features used to control terminals and programs. The Operator Communications (OC) task, Job Control, and other programs can be entered through TMP.

A user can access TMP by pressing the BREAK key or by pressing the CONTROL key and the A key simultaneously (CONTROL A sequence). This action is called an attention request. In addition to providing a set of directives for task control, TMP supports the control of Ring Detect communications lines.

TMP detects attention requests, processes commands, and sends responses for several terminals simultaneously. When an attention request is made, TMP takes exclusive use of the terminal. This enables the user to interrupt a program in order to control its execution. TMP does not process commands itself, but calls in an overlay to do the work.

2.2.4 OPERATOR COMMUNICATIONS (OC) TASK

The Operator Communications (OC) task provides on-line control of MAX IV operations. The operator can assign tasks to priority levels, control the execution of tasks, add control commands, and perform many more functions.

The Operator Communications task may be accessed from any terminal connected to the system, using the Terminal Monitor Program. A password protection scheme is provided to limit unauthorized access. In addition, terminals have access to a subset of the full OC package to provide local task control.

Operator Communications directives may be made memory-resident in order to increase speed or to allow an operator to communicate with the system that does not have a disc (for example, a system that has been loaded from a remote site).

Nonresident OC directives are loaded as overlays from disc while resident directives reside in the MAX IV OS area of memory. Whether nonresident or resident, these directives utilize identical program modules.

This resident system task operates at a high task priority level to ensure that it has control over all tasks that need its services. Generally, this task's execution is suspended at all times except when the operator has requested to communicate with the system or when the system's Exceptional Condition (X) Task has informed the operator of exceptional conditions that require response.

2.2.4.1 Using the OC Task

The task is activated manually (actually it is aborted and restarted) by pressing the CONSOLE INTERRUPT switch on the Control Panel or by pressing BREAK or CONTROL A on the operator's terminal. It identifies itself, queues up a READ to the operator's terminal, and then usually relinquishes time to all lower priority tasks until the first character of an operator directive (command sentence) is typed by the operator. As an option, the OC task can be implemented so that if 10 seconds expire without such a response, the READ terminates and the task suspends itself.

If the system must type emergency information to inform the operator, the Exceptional Condition Task queues the message and internally requests the operator task if its response is necessary. If the operator task is already active because it is waiting for an operator entry or is performing a directive function unrelated to the emergency, it is aborted and restarted after a suitable time delay so that the emergency condition(s) can be reported. After the first character of a directive is received, the task takes exclusive-use of the terminal device until the next READ Directive is queued or when the task suspends itself.

When a directive is received from the operator, the task examines the first keyword of the command sentence to see if it matches an entry in a nonresident directory. If the desired operator communications directive processor is found in the nonresident directory, it is loaded as an overlay into a fixed buffer and executed. Long directive processors may consist of two or more loaded and executed overlay routines. Control then returns to the operator so that a new directive may be entered.

If a new directive is not received within the time limit, or if the directive is a terminating directive, the operator task will be suspended. The execution of some directives causes the operator task to become inactive (suspended immediately after they are executed) requiring that the operator again press the CONSOLE INTERRUPT switch on the Control Panel or BREAK or CONTROL A on a terminal device when the services of this task are needed. Such directives are called terminating directives.

The Operator Communications Task is described fully in the MAX IV OPERATOR COMMUNICATIONS, System Guide, listed in the Preface.

2.2.4.2 Adding Custom Operator Directives

A user can add custom operator directives to the operating system as long as the first keyword (up to three characters) of the directive is unique. Such directives may be programmed to address the currently attentive task, the system as a whole, or a particular task only.

These directives may be added to the operating system at run time by simply cataloging a series of one or more short overlay programs in a special disc file (SM). The first or root overlay is cataloged in this "system module" file under the name of the 3-character keyword used to start it. Secondary overlays may be cataloged under 3-character program names that have a dollar sign ($) as the second character. These overlays operate in MAP 0 in the privileged mode.

2.2.5 JOB CONTROL (JC)

Job Control (JC) is that part of the operating system that provides facilities for both multiple batch job streams and multiple interactive terminals for software development. Both types of processes are monitored by Job Control as it interprets Job Control directives supplied serially by a Command Input terminal or card reader.

For detailed information, refer to the MAX IV/MAX 32 NONRESIDENT JOB CONTROL AND BATCH FACILITIES, Programmer's Reference Manual, listed in the Preface. Refer to Figure 2-3 for an illustration of the logical files used by Job Control.

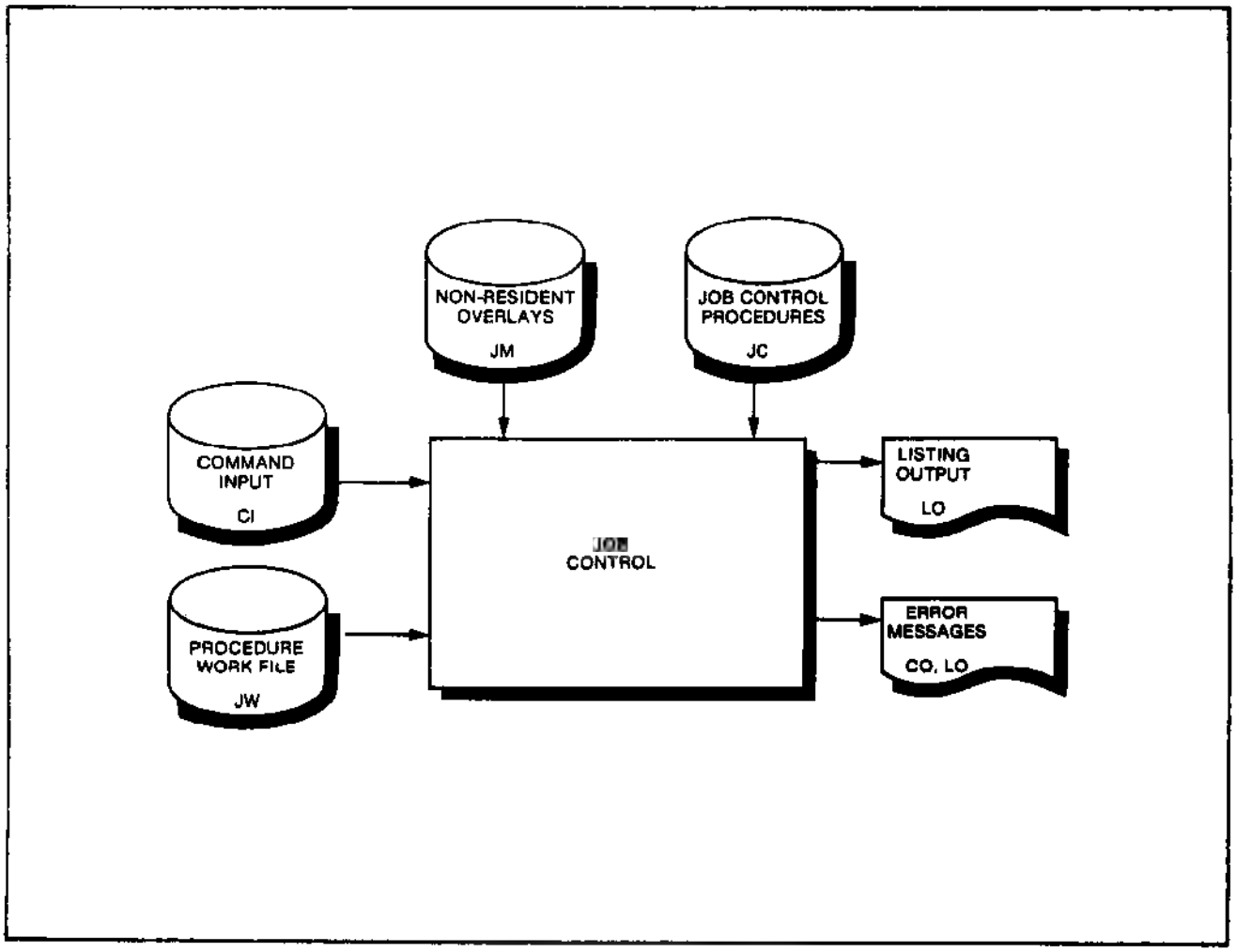


Figure 2-3. Logical Files Used by Job Control

2.2.5.1 Job Control Language

The Job Control Language is a very flexible procedural language that allows the user to control the execution and system resources of system processors such as the FORTRAN Compiler or any user-written program. User-written programs can be compiled, assembled, link-edited, cataloged and executed through an appropriate stream of Job Control directives. The resulting user programs can be cataloged as real-time tasks.

2.2.5.2 Job Control Procedures

One of the most powerful features of Job Control is the ability to define a series of Job Control statements (and other batch-processor directive statements) as a procedure. Once defined, a procedure has the characteristics of a control language subroutine (or "macro") that can be called upon by name. These procedures can even have variable arguments (parameters) passed to them. Procedures can call other procedures to any level of nesting and have conditional looping and testing facilities.

With this feature, a library of frequently used Job Control procedures can be cataloged on disc for instant access by call. This is of considerable value when people with a wide variation in experience levels must share the resources of the computer system. Inexperienced users can be taught to use only a limited set of convenient (but restrictive) procedures, which limit their access to the machine.

2.2.5.3 Batch Processing Under Job Control

The root task body (or initial overlay) of any standard batch-processing task is the nonresident Job Control program. This program acts as the original job initiator and establishes the initial memory and file resources for the first job step. It then overlays itself with the loading of the batch-processor program defined by the first job step. The Job Control program is also automatically loaded between all job steps so that it can establish appropriate memory and file resources at each step. The next job step is invoked by the current job step calling the EXIT service. Thus, the Job Control language facilitates the sequencing and resource control of all job steps through the termination of the job. At the end of a job, the batch-processing task seeks a new job or suspends itself if no suitable job can be found.

Batch-processing tasks take all their directions from job streams. Job streams are sequential files of data consisting mostly of Job Control Directives or procedures. The user's data to be processed can also be embedded in job streams or located on other data files. The job stream may be considered to be endless and consisting of many independent and unrelated jobs. Each job consists of one or more job steps. Job steps are related to each other and must be performed sequentially as entered. Individual jobs are assumed to be unrelated and may be executed (at the system's choice) in any order -- depending upon size, priority, and availability of resources.

2.2.6 TASKMASTER

In the MAX IV multiprogramming/multitasking environment, many tasks may compete for CPU time. Order of execution is maintained by the Taskmaster. The Taskmaster is the resident system element that transfers CPU control to the highest priority task ready to work. Task priority levels are assigned to tasks below the lowest hardware level. There may be up to 256 unique software priority levels, with multiple task assignments to a single priority level. Waiting tasks (such as I/O-bound or interactive tasks) are given timely response, and compute-bound tasks are given access to the CPU at regular intervals.

2.2.7 MODULE LOADER (ML)

Nonresident overlays and tasks are cataloged on disc in a special load module format. This quick-load format allows the MAX IV System to map and load a task or overlay near the speed of the bulk storage device (moving head disc). Once a task's memory is allocated the load operation can proceed into the contiguous virtual memory space of the task defined by the task's map image.

The load module is in a format that requires no software relocation if the program has been cataloged in a normal manner with the MAX IV Task/Overlay Cataloger. The Module Loader utilizes the natural hardware relocation benefits of memory mapping. The virtual locations of all nonresident programs, global commons, and re-entrant shared libraries are determined at link-edit time or cataloging time. For special applications, a module may still be cataloged so that it can be relocated at load time, but this is rarely necessary in a MAX IV system.

Each load module of a root task or overlay consists of two types of records called Load Program Records (LPRs) and Module Object Records (MORs).

2.2.7.1 Load Program Records - LPRs

A Load Program Record contains information about the resources for the program to be loaded. The system uses this information to select the appropriate memory and space environment of the program to be loaded (if not already allocated). For a root task, the LPR also contains the task's resource assignments that tell the Module Loader how to build a Task Control Block (TCB) for the task when it is activated and becomes an independent multiprogramming/multitasking entity.

The LPR also contains a "string chain" that specifies to the loader how it should read the body of the program. This chain facilitates the "scatter" read of the program even when non-contiguous in memory. The chain is in a format that is directly interpretable by the Input/Output Processor of the CLASSIC computer.

2.2.7.2 Module Object Records - MORs

The body of a nonresident program consists of one or more program extents called Module Object Records that are defined in the LPRs. These MORs are read into the program's allocated memory area using a scatter-read technique as specified by the string chain in the LPRs. Only one disc access is normally required to complete the transfer.

2.2.7.3 Resident Load Module Directories

All load module files have their directory of cataloged programs as the first data item on the file. For critical program storage files, a resident directory may be defined in the system, which keeps resident copies of all or part of the file's load module directory for immediate access by the Module Loader.

2.2.8 OUTPUT SPOOLING (S) TASK

Output spooling is a technique that allows one or more output devices to serve the needs of many concurrent batch-processing and real-time tasks. The technique makes a program use a high-speed disc file as a buffer so jobs can complete their processing quickly and exit even though the physical output device has not yet received the information.

Spooling serves to increase throughput by keeping devices busy and by isolating processing from manual interventions (such as paper changing and device loading).

The Output Spooler is actually an optional symbiont task operating at a system task level. This task routes output device requests to the disc by its own memory buffers and eventually outputs the disc resident information to the desired target devices. The Output Spooling Task supports printing devices. Printing information is compressed/uncompressed as it is transferred to/from disc in order to reduce the number of disc accesses and to save disc space.

2.2.9 EXCEPTIONAL CONDITION (X) TASK

The Exceptional Condition (X) Task is a resident system task that operates at the highest task priority level (0). Its purpose is to report exceptional system conditions to the system operator and to perform task-level operations for the MAX IV OS itself. When necessary, the X task:

- Prints various exceptional condition reports requested by direct interrupt driven elements of the system that cannot queue messages themselves.
- Reports device and memory failures and off-line conditions to the system operator.
- Reports tasks that exceed their cataloged time limits.
- Performs system start-up and initialization functions including the initialization of resident directories for disc files.
- Performs user-coded system restart procedures.
- Interrupts operator directive entries if important messages must be typed on the operator's terminal.
- Updates calendar tables.
- Performs the first phase of the volume recognition sequence.
- Updates the Task Queue if the ROUNDROBIN scheduler is optioned.
- Performs Type Zero disc data structure updates.

2.2.10 INTER-TASK COMMUNICATIONS

The MAX IV Inter-Task Communications (ITC) message service is a system module that allows for variable length collections of data to be transmitted between tasks. Messages flow along distinct communication paths known as Virtual Circuits (VC's). When a VC has been opened by agreement between two tasks, both tasks may send and receive messages along this bidirectional path. A task may have multiple VC's open between itself and other tasks, and may even have several distinct VC's open between itself and a single other task. A VC may at anytime be cleared by either task, thereby ending the communication.

Optionally, a task waiting for information that has not yet been transmitted can suspend itself until the message arrives. Thus, tasks may use the message service for synchronization as well as information transfer.

Operator Control directives can be used to dump information stored in various message service data structures, generating information about messages queued on one or more VC's.

2.3 SYSTEM GENERATION

The system generation process requires the user to write several specification statements in a higher-level MACRO language. These statements will permit the user to customize the MAX IV system, including only those optional elements that are required for applications. System generation is described in the MAX IV SYSGEN, System Guide Manual, listed in the Preface.

Generation of the memory resident elements of an operating system is accomplished using the MAX IV MACRO Assembler (M5A) and the MAX III Link Editor. The Stand-Alone Linking Loader may be substituted for the MAX III Link Editor in cases where the linking time can be tolerated each time a fresh copy of the system must be loaded into main memory. The nonresident (disc resident) elements of the operating system are prepared on disc with the MAX IV Task/Overlay Cataloger and other utilities.

A system generation is performed in four phases:

PHASE 1 Specify and assemble the desired resident elements in the following four parts with a MACRO statements language:

a) System Structure Elements
b) Basic I/O System Elements
c) Task Related Elements
d) File Manager Configuration

To these special assemblies, other user-specified elements may be supplied (resident REX Services, operator directives, resident task programs). The resulting group of object programs may be thought of collectively as the "main program" of the operating system. In most cases, these program elements consist of only simple data bases and linkages that "call" the appropriate elements from the MAX IV system elements library.

PHASE 2 Link the object elements produced in Phase 1 with the elements of the MAX IV system elements library, loading only the routines that were referenced.

PHASE 3 Load and initialize the system object image in main memory. A special Stand-Alone Loader (SAL) may be used to perform this function.

PHASE 4 Use the newly generated system to establish all nonresident elements.

For detailed information, refer to the MAX IV SYSGEN, System Guide, listed in the Preface.

2.4 STAND-ALONE SOFTWARE

Stand-Alone Support Software is a collection of programs for various booting, copying, dumping, and initializing functions. These programs run in a stand-alone mode, that is, without the need of the operating system. Refer to the MODCOMP STAND-ALONE SUPPORT SOFTWARE, Programmer's Reference Manual, listed in the Preface, for detailed information on each program.

Among the frequently used Stand-Alone Support Software programs are:

- Disc/Tape Copy
- Stand-Alone Linking Loader (SAL)
- Alternate Track Initializer

The Disc/Tape Copy Program duplicates the contents of disc packs by copying the contents from a disc medium to another disc medium of the same disc type. In addition, the contents of a disc pack can be copied to a reel of magnetic tape and then copied back to a disc pack. All models of discs and magnetic tape drives are supported.

The Stand-Alone Linking Loader (SAL) contains the appropriate stand-alone handlers in order to load variable length binary records from any suitable standard device. SAL is available in two forms. The first form is relocatable. The second form is fillable and can relocate a copy of itself in upper memory automatically.

The Alternate Track Initializer is part of MODCOMP's method of recognizing disc defects and re-assigning track locations to other areas of the disc. This stand-alone program reads, reformats, and relocates the vendors bad track map into the MODCOMP Alternate Track Map. In addition, the Alternate Track Initializer can:

- Add or delete alternate assignments from the MODCOMP Alternate Track Map.
- Create a new MODCOMP Alternate Track Map by surface testing the disc media.
- Write-prep the disc pack.

2.5 MAGIC

MAGIC is the Multi-User Applications and Guide for Inquisitive Customers and consists of software products designed to aid the system analyst in building a unique multi-batch system. The features of MAGIC include:

- Electronic Mail
- Electronic Bulletin Board
- Source Compare
- Text Formatter
- Online HELP
- Sign-On Task
- Pre-Scheduled Task Program
- MAGIC Manual

MAGIC requires the system to have at least one File Manager disc available.

The following sections provide general summaries of these features. Detailed descriptions can be found in the MAX IV MAGIC, Programmer's Reference Manual listed in the Preface.

2.5.1 ELECTRONIC MAIL

The MAX IV Electronic Mail Processor (MAIL) provides a mailbox facility whereby the user can send and receive letters from other users on a multi-batch system. The user's mail is listed chronologically in the directory, with a number assigned to the individual letters. The letters can be examined by calling up the number assigned to that letter.

Electronic Mail allows the user to:

- o display or print a letter on the mailbox
- o send "registered" mail (a receipt is issued to the sender when the recipient reads the letter for the first time)
- o access the list of individuals/groups capable of receiving mail
- o delete a letter from the mailbox
- o send a letter to another mailbox - individual or group, where everyone in the group receives a copy of the letter
- o forward a letter to another user
- o reply to a letter
- o mark a letter as "unread"

2.5.2 ELECTRONIC BULLETIN BOARD

The Electronic Bulletin Board Processor (EBB) provides a general information service to all users. Information can be added, deleted, or displayed (printed) by any user. Each entry is categorized by topic, and within each topic the items are sorted chronologically, with the newest item appearing first.

2.5.3 SOURCE COMPARE

The Source Compare Processor (CMP) allows the user to compare two modules and list the differences. This feature makes Source Compare an aid in detecting contrasts in source or text.

To use Source Compare, the program CMP must exist on the binary load module file, BM, and the Job Control procedure COMPARE must reside on the Job Control procedure file, JC. Also, the modules to be compared must reside on directorized partitions, such as USL.

Each line of the two modules are compared, and any line that is different is listed, along with the next sequential line, in the line output device.

2.5.4 TEXT FORMATTER

The Text Formatter Processor provides an interface between man and machine to simplify the development of consistent alphanumeric input/output through CRT terminals.
The Text Formatter provides subroutines in the system library to handle the details of text processing.

Text Formatter input consists of command lines and text lines. The command lines control how the text lines are processed and output. Some capabilities of the Text Formatter commands include:

- setting boldface text
- setting page footer both even/odd
- setting page header both even/odd
- breaking for new page
- setting left/right margins
- setting line spacing
- setting page length
- setting number of lines above/below the header/footer
- setting next page number

2.5.5 ONLINE HELP

The Online HELP tool is designed to aid the customer on a multi-batch system. The HELP service is a series of TMP (Terminal Monitoring Program) overlays and, since it runs in TMP, HELP is accessable to the user at all times. The user can also exit HELP (and TMP) and be returned to the place in process before entering HELP.

The information contained on the HELP files is instructive, providing the user with short discussions of various system processors, error messages and other features of the user's system.

A HELP screen prints up to 18 lines of information at a time and a PAG command allows the user to skip pages or revisit previously viewed pages.

HELP files can be added, removed, or changed by locating the USL directory name where the HELP files are located and by using Source Editor (SED) or MODCOMP Screen Editor (MSED). In this way, users can create their own files custom-tailored to their applications.

2.5.6 SIGN-ON TASK

The Sign-On Task provides information, programs, and procedures for building a log-on method for users of a multi-batch system. The Sign-On and Set-Up Tasks perform the following functions:

- check user passwords
- refresh the screen with a message and the current time/date
- open user files
- assign user files
- check for Electronic Mail
- activate interactive batch tasks

The Sign-On source code is provided with instructions to modify the code to suit individual user needs.

2.5.7 PRESCHEDULED TASK PROGRAM

The Prescheduled Task (PRETSK) procedure activates other tasks. PRETSK allows the system manager to add or decrease the number of prescheduled tasks without changing the SYSGEN. Instead of changing the SYSGEN, the prescheduled tasks are added at the end of the PRETSK procedure. PRETSK must be in the SYSGEN as a prescheduled task and the task must be reassembled and placed on the appropriate Task Overlay Cataloger (TOC) file.

2.5.8 MAX IV MAGIC MANUAL

The MAGIC Manual is designed to aid the user in building a multi-batch system that best fits the user's needs. Some of the topics included in this manual are:

- reducing MAP 0 requirements
- building user and batch files
- dual porting
- adding a batch task to a system
- installing the MAX IV Printer Spooler/Despooler Subsystem in a SYSGEN
- creating a memory-resident batch module directory
- SYSGENing File Manager
- adding the Dump Analyzer package to a SYSGEN

Additional information on each of the previously discussed processors and programs of MAGIC is also included in the manual, along with procedures and example programs to guide the user in designing a unique system.

CHAPTER 3
THE BASIC INPUT/OUTPUT SYSTEM

The Basic Input/Output System (BIOS) of the MAX IV Operating System is discussed in this chapter in terms of:

- o an overview of BIOS
- o the components of BIOS
- o the handler types with the BIOS

3.1 BASIC INPUT/OUTPUT SYSTEM (BIOS) OVERVIEW

The Basic Input/Output System (BIOS) is a common central routine to which all requests for I/O operations are directed. BIOS schedules and monitors all I/O transactions to the peripheral controllers or symbiont tasks. BIOS is re-entrant; that is, it may be interrupted at any point of one task-level execution and be immediately called by another task (re-entered).

BIOS is a logical input/output system. That is, a program does not directly access an I/O device. All I/O operations are directed to previously assigned logical files. A logical file is a name that points to a real device. Assignment of logical file names to target devices may be performed at system generation, at task cataloging, or at run time by using Job Control commands or logical assignments within the task itself.

Input/Output device handlers are written in a "device independent" manner. This minimizes the need for special considerations for the actual device used at run time. There are, however, options that may be specified when the need for a "device dependent" operation arises.

BIOS allows levels of queuing to common or separate devices. I/O requests are queued in order of the calling task's priority and chronologically for tasks of equal priority. Either of two error recovery techniques may be chosen when requesting a BIOS service - system error recovery or user error recovery.

BIOS also supports a data chaining mode of operations for privileged programs. This mode performs an I/O operation that writes a continuous record on an output medium consisting of buffers gathered from non-contiguous areas of memory; however, data chaining is limited to mass storage devices (i.e., disc and magnetic tape).

For detailed information, refer to the MAX IV BASIC I/O SYSTEM, System Guide, listed in the Preface. Refer to Figure 3-1 for an illustration of the parts of the MAX IV BIOS.

3.2 COMPONENTS OF THE BASIC INPUT/OUTPUT SYSTEM

The MAX IV Basic I/O System (BIOS) consists of two parts:

- o Interrupt-driven or DMP-driven handlers exist for all physical I/O devices. These handlers are directly-connected to the I/O channels they control. The user need not interface to them directly.

- A centralized logical Basic I/O System, addressable as a series of Executive (REX) Services calls, allows a user to interface to device handlers indirectly - by way of logical files. Logical files are assigned to devices at task cataloging time, or at run time. These services permit a programmer to talk to the file's assigned device only in terms of physical records (for example, card, block, or sector).

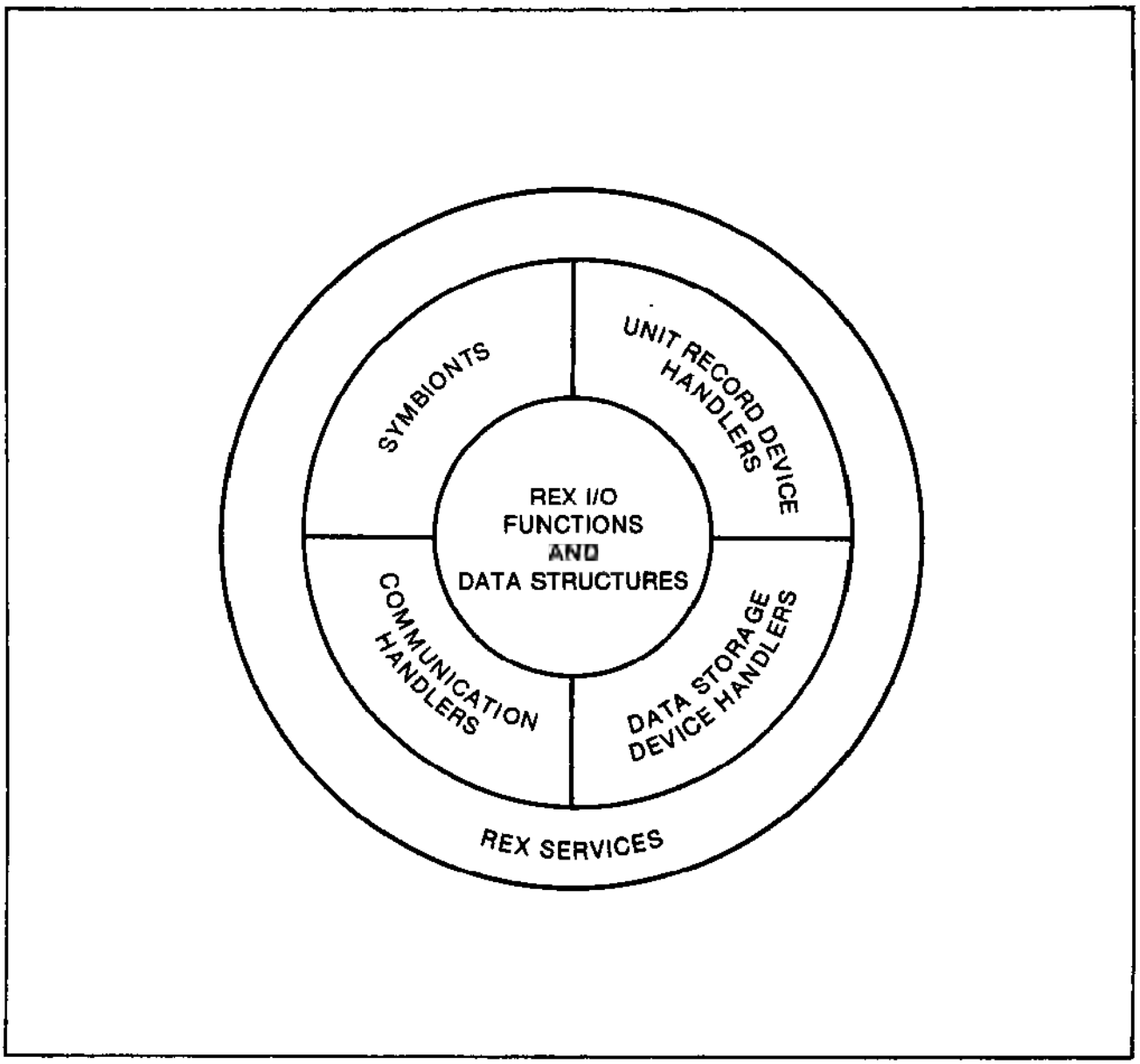


Figure 3-1. The MAX IV Basic Input/Output System

3.2.1 I/O HANDLERS AND DEVICES

Devices may be one of the following:

- o Real devices, for example, a line printer or complete disc.
- o Logical devices, for example, subdivisions of a disc medium into real files.
- o Imaginary, for example, simulated by a task called a symbiont. Refer to section 3.4.

Logical files can point to real, logical, or imaginary devices.

Some devices have their own independent controller channel, while others must share a common controller channel. All devices are identified by name in the Logical Device Table. The interrupt-driven handlers permit I/O operations to be performed concurrently (overlapped) with the execution of the programs that request the operations. Slow devices use data interrupts to transmit data bytes/words; higher speed devices use Direct Memory Processors (DMP's) to reduce interrupt overhead. Service interrupts are used by all devices for initiating and terminating all operations. These interrupts manage the queue of operations waiting to be started on a particular controller channel.

Device error recovery may be automatic (retry initiated by service interrupt with unrecoverable errors reported to the system operator) or may be under the complete control of the calling task. The latter method allows a critical task to perform suitable application-dependent error recovery techniques, including device substitution.

Device handlers have both device-independent and device-dependent modes of operation. The former mode lets the handler simulate (if necessary) a series of device characteristics that are common to a class of similar (but not identical) devices. The latter mode allows a programmer to use the unique characteristics of a particular device if desired.

3.2.2 BASIC I/O SYSTEM SERVICES

The user calls the I/O system by a series of Executive (REX) Services that all converge on a large central scheduling routine. This routine executes at the calling task level until an I/O operation is scheduled in the queue of the appropriate I/O device controller. These services schedule such functions as READ, WRITE, REWIND, and WEOF.

Each task has a local logical file list called the task's File Assignment Table (FAT), located in the Extended Control Block (EXT). MAX IV has a similar list of logical files, called the Global File Assignment Table (GFAT), that is kept in the EXT of the standard Exceptional Condition (X) task. These tables are used by the Basic I/O System to find the assigned device when an I/O operation is requested for a particular logical file by a program. The logical files exist only as linked entries in these lists. The task's File Assignment Table (FAT) is always used unless the desired file cannot be found in it. In this case, the Global File Assign Table (GFAT) is used. With this scheme, duplicate logical files may exist for different tasks -- and yet unique device assignments can be maintained. Likewise, a single logical file name can be used for communications with a device by many different tasks.

Logical files may be assigned to other logical files, but the final assignment must be to a device (real, logical, or imaginary) that must be in the System Logical Device Table. Disc devices (not files) may be kept permanently in this table or may be added and deleted with File Manager services, Job Control Directives, or special batch-processing utility programs.

Each logical file entry has two assignments called the current assignment and the default assignment. With this feature, a task may change its file/device assignments at run time and restore original assignments when desired. This is particularly useful for batch-processing tasks, where each new job can be given pre-set standard file assignments by using all the default assignments.

Each logical file has a name. The programmer can address the file only by its name.

All I/O operations that cannot be started immediately are queued so that no program delay results due to a busy controller channel. Operation nodes in device controller queues are ordered by calling task priority and by chronological order for like task priorities.

Devices that share a common controller channel have "slough" queues that prevent unmounted or unavailable devices from stalling the central controller channel queue.

When using the Basic I/O System directly, the user must expect to deal with the assigned device in terms of discrete physical records or blocks. Buffers must be supplied in the user's program that are capable of containing exactly one physical record. Each device has a different physical record size; therefore, the user must often determine device size and adjust the program accordingly. The largest common buffer size for most standard devices does not exceed 128 words (1/2 memory page), although some special devices may be programmed with much larger buffers if the program is capable of using memory resources to such an extent.

Data is transmitted to and from memory data buffers within the program body of the calling task. In a FORTRAN program, which uses subroutines to define access methods, these data buffers may be transparent to the user; but are allocated and used by the FORTRAN Run-Time Package and reside in a labeled common area generated by the FORTRAN IV compiler or in an area near the end of the task body.

3.3 INTERRUPT-DRIVEN PERIPHERAL DEVICES

Many standard I/O devices are supported by the MAX IV Basic I/O System. These include:

- Interactive Terminals (CRTs and hard-copy devices)
- High-Speed Line Printers
- Slow- and Medium-Speed (one character at a time) printers
- Raster-Type Plotter/Printers
- X-Y Plotter
- Magnetic Tapes (9 track, NRZI and phase-encoded)
- Floppy Discs
- Cartridge Moving Head Discs
- Multi-Platter Moving Head Discs
- Large-Capacity Moving Head Discs
- Fixed Head Discs
- Memory Plus (simulates a Fixed Head disc except uses memory for storage)
- Communication Channels (asynchronous and synchronous)

3.4 SYMBIONTS

Unusual devices, which have difficult or unusual control protocol or perform functions unavailable in real devices, may be interfaced to the system by a symbiont. These symbionts are simply tasks that present an imaginary device interface to the programmer while insulating the user from the peculiar protocol of the real device or communications channel. Typical symbionts include:

- Output Spooler
- Remote CPU Interfaces (such as 2780/3780)

3.5 HANDLERS

The handlers contained in the MAX IV BIOS are categorized as follows:

- Unit Record Device Handlers
- Data Storage Device Handlers
- Communication Handlers

3.5.1 UNIT RECORD DEVICE HANDLERS

Unit Record Device handlers support devices such as console keyboards, card readers, line printers, and spooling symbionts. For detailed information, refer to the MAX IV UNIT RECORD DEVICE HANDLERS, System Guide, listed in the Preface. The Unit Record Device handlers are:

- Half-Duplex Teletypewriter/CRT
- Line Printer
- Raster Printer/Plotter
- Incremental X-Y Plotter
- Imaginary Printer Spooling Symbiont
- Input Spooling Symbiont

3.5.2 DATA STORAGE DEVICE HANDLERS

Data Storage Device handlers support devices such as magnetic tape, bulk memory, and many types of discs. For detailed information, refer to the MAX IV DATA STORAGE DEVICE HANDLERS, System Guide, listed in the Preface. The Data Storage Device handlers are:

- Magnetic Tape
- CLASSIC Disc Device
- Fixed and Moving Head Discs
- Large Capacity Moving Head Discs
- Bulk Memory

3.5.3 COMMUNICATIONS HANDLERS

Communications handlers support interfaces in such environments as local or remote asynchronous data terminals, SDLC/HDLC communications channels, and Process Control Input/Output devices. For detailed information, refer to the MAX IV COMMUNICATION HANDLERS, System Guide, listed in the Preface. The communications handlers are:

- Asynchronous
- Bisync
- SDLC/HDLC Frame
- Standard Link Frame Level
- Computer Link/Data Terminal
- Local Process I/O

CHAPTER 4
FILE MANAGER SUBSYSTEM

This chapter presents an overview of the MAX IV File Manager Subsystem, a system generation option of the MAX IV Operating System. A discussion of the subsystem and various File Manager system processors are included.

4.1 A DEFINITION OF THE FILE MANAGER SUBSYSTEM

At system generation, the user can elect to use the File Manager Subsystem to monitor and control disc usage. The File Manager organizes, maintains, and services multi-level files. Named data files and file directories can be nested, with facilities for volume and file security.

The File Manager concerns itself with the definition and use of collections of physical records (files) and the utilization of space on secondary storage volumes.

For detailed information, refer to the MAX IV FILE MANAGER, File Management Manual, listed in the Preface. Refer to Figure 4-1 for an illustration of the parts of the File Manager.

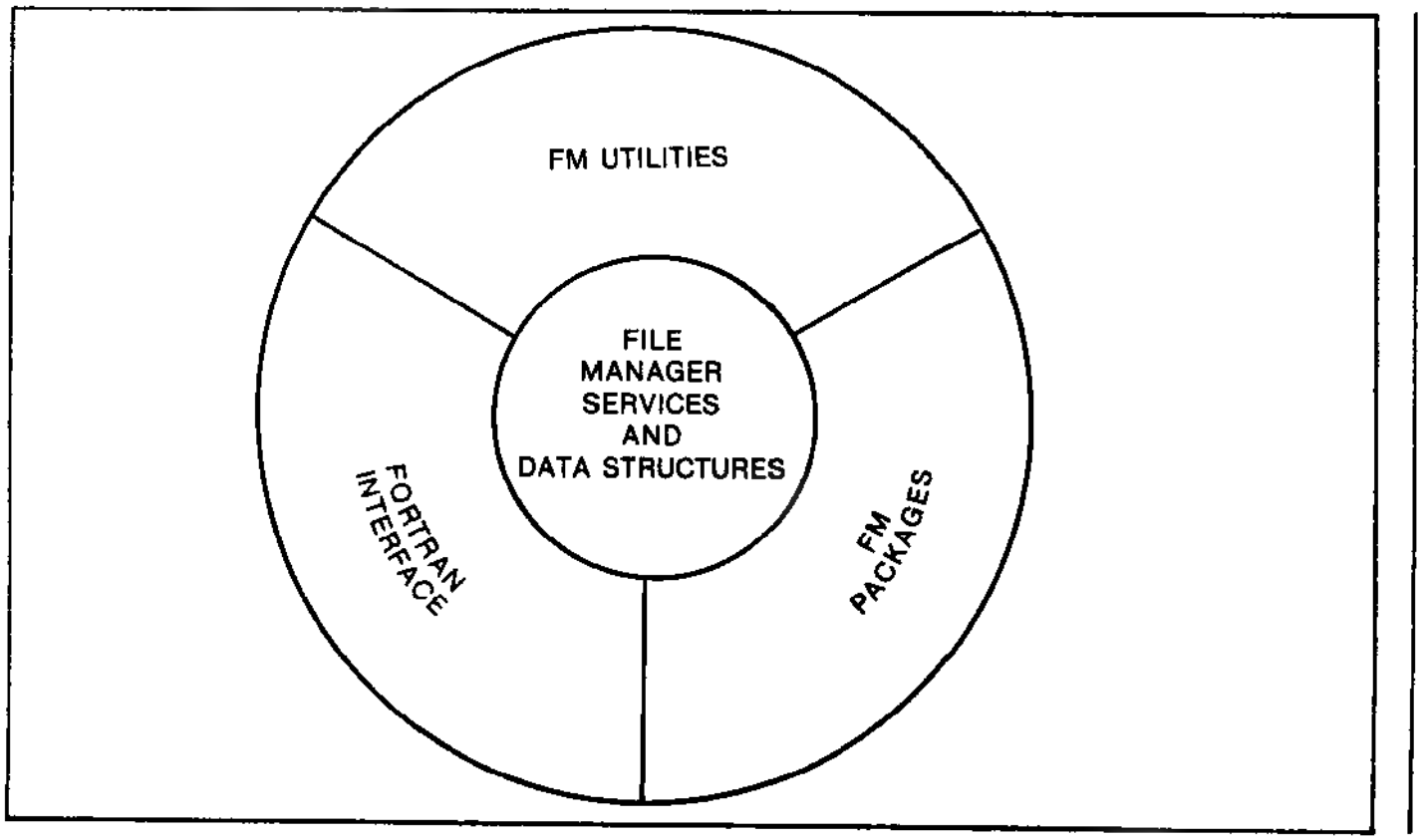


Figure 4-1. MAX IV File Manager

4.2 FUNCTIONS OF THE FILE MANAGER SUBSYSTEM

The File Manager Subsystem provides services to:

- o Define and create files in a hierarchical file system
- o Permit access and usage of files by tasks
- o Control file access and usage

The File Manager functions allow a task that has associated programs and files that were defined using the Basic I/O System of the MAX IV OS to operate correctly with the File Manager Subsystem with no change of file structures or service calling sequences. Any task may use File Manager services with the Basic I/O System services without conflict.

4.3 FILE ORGANIZATION

File organization is discussed in terms of:

- o File hierarchy
- o File identification
- o File record organization

4.3.1 FILE HIERARCHY

The File Manager Subsystem enables organization of files into hierarchical levels. Each level divides the media associated with a file into individually accessible files. This capability permits the user to define families of files. The level at which a file exists in the hierarchy is defined by the user when the file is initially created. The File Manager Subsystem defines all files as Directory Files or Data Files.

A Directory File is a file that maintains entries describing the existence and location of files defined at the next lower level in the hierarchy. Once a file is used as a Directory File, it may not be used as a data file until it is removed from the system and recreated as a data file.

A Data File is a file that contains data for storage and manipulation in a format defined by the user of the file. A Data File is always pointed to by a Directory File. Once a file is used as a Data File, it may not be used as a Directory File until it is removed from the system and recreated as a Directory File.

The File Manager Subsystem can also create partition data files. Partition data files are File Manager files that look like standard BIOS transient disc partitions. This type of file must conform to the following four rules:

- o It must be a first-level file.
- o It must have a 3-character Compressed Alphanumeric (CAN) codable name.
- o Its file space must be wholly contiguous.
- o The initial file size must be equal to the maximum file size.

The user can address this type of file as though the file were maintained by BIOS.

4.3.2 FILE IDENTIFICATION

The primary identification of a file is the file name. A file name is defined by a string of bytes supplied by the user when the file is created. The file name uniquely identifies the file and specifies the file's hierarchical position.

A file in the highest hierarchical level of the file system is identified by a simple (single) file name. Any file in any other hierarchical level must be referenced with a compound file name (two or more simple file names).

4.3.3 FILE RECORD ORGANIZATION

A file is defined as a named collection of physical records. These records are separated into two distinct groups based on their usage:

- o File Management Information Records
- o File Data Records

The File Management Information Records contain information necessary to define and control the file. The information contained in these records consists of three sets of items:

- o File Control Information (the File Label)
- o File Space Control Information
- o File Directory Information

All other physical records of a data file are defined as File Data Records. The organization, content, and format of these physical records are defined by the user. These records are read and written by the standard BIOS READ/WRITE services. Directory Files contain no File Data Records.

4.3.4 FILE MANAGER PACKAGES

Within the File Manager Subsystem, three optional packages exist that are specified during SYSGEN. They include:

- o The Security package
- o The Catalog package
- o The Audit package

The Security package has two purposes:

- o The package determines if the requesting user is allowed to access the target volume or file.

- o The package determines what type of access the user is allowed.

The Catalog package references a disc file on which information concerning file residency is maintained. Therefore, the user can generate a File Descriptor List that contains the file name and the Catalog package supplies the volume and transport names.

The Audit package provides an audit trail of all or selected File Manager operations. The Audit package can track such items as time of day, date, volume name, and so forth.

4.4 FILE MANAGER VOLUME PREPARATION (FMPREP) UTILITY

The File Manager Volume Preparation (FMPREP) Utility performs the following functions:

- Preps a new disc volume for use as a File Manager volume.
- Verifies the usability of the disc volume media.
- Constructs and records the File Manager's Volume Management Information on the volume.

For detailed information, refer to the MAX IV FILE MANAGER, File Management Manual, listed in the Preface.

4.5 FILE MANAGER LISTING (FMLIST) UTILITY

The MAX IV/MAX 32 File Manager Listing (FMLIST) Utility operates as a standard system processor within the File Manager Subsystem of MAX IV. FMLIST produces reports that list the names and characteristics of files contained on File Manager volumes, as well as the characteristics of the volume itself. In addition, reports that reflect files within certain individual directories or single files can be generated.

FMLIST also features a report type that allows a hexadecimal dump of the file management records read by FMLIST. The report enables a manual diagnosis of problems. The hexadecimal dump is printed only if the user has entered the correct security parameters. This is used to prevent unauthorized access to system files.

For detailed information, refer to the MAX IV/MAX 32 FILE MANAGER FILE LISTING (FMLIST) UTILITY, Programmer's Reference Manual, listed in the Preface.

4.6 FILE MANAGER SAVE/RESTORE (FMSAVE) UTILITY

The MAX IV/MAX 32 File Manager Save/Restore (FMSAVE) Utility operates as a standard system processor within the File Manager Subsystem of the MAX IV Operating System.

FMSAVE copies files from a File Manager volume to magnetic tape (SAVE) and from magnetic tape to a File Manager volume (RESTORE). In addition, listings can be produced that reflect the contents of the magnetic tape (LIST).

FMSAVE can run as an interactive program and as a batch program. An interactive HELP function is provided that displays the syntax and valid parameters of directives.

4.6.1 SAVE FUNCTION

The SAVE function copies file contents and header information from a File Manager volume to magnetic tape. The SAVE function can be used in the following ways:

- A complete File Manager volume, including all directories, partitions, and data files on the volume, can be saved (copied) to magnetic tape.
- A complete directory, including all directories and data files that it contains, can be saved to magnetic tape.

- An individual data file or partition can be saved to magnetic tape.
- Directories and data files specified in a logical file can be saved to magnetic tape.
- All data and directory files that have been updated on or after a certain date can be saved to magnetic tape.

Multiple SAVE operations can use the same magnetic tape. Therefore, one magnetic tape can contain the saved copies of several directories. In addition, FMSAVE can process a SAVE operation that requires multiple magnetic tapes, for example, saving large disc volumes.

4.6.2 RESTORE FUNCTION

The RESTORE function copies file contents and header information from magnetic tape to a File Manager volume. The RESTORE function can be used in the following ways:

- A complete saved File Manager volume, including all directories, partitions, and data files on the volume, can be restored (copied) to the original volume or to a different volume. When restoring a complete volume to the original volume, all first-level files on the original volume are destroyed before the RESTORE operation is performed.
- A complete saved directory, including all directories and data files that it contains, can be restored under the original directory name or under a different directory name.
- Part of a saved volume or directory can be restored.
- A saved data file can be restored under the original data file name or under a different data file name.
- Directories and data files specified in a logical file can be restored.

If the data file or directory being restored does not exist, it is created. If the data file or directory being restored does exist, the data file or directory on the volume is destroyed and the data file or directory on the magnetic tape is restored to the volume.

4.6.3 LIST FUNCTION

The LIST function produces a listing of the files contained on a backup magnetic tape. While this listing is produced automatically by the SAVE and RESTORE functions, the explicit LIST function is provided as an additional means to obtain a listing of the contents of an existing backup tape.

The listing produced automatically by the SAVE and RESTORE functions indicates any errors that occurred during the operation. If possible, FMSAVE continues with the operation even if errors are found.

For detailed information, refer to the MAX IV/MAX 32 FILE MANAGER SAVE/RESTORE (FMSAVE) UTILITY, Programmer's Reference Manual, listed in the Preface.

4.7 FORTRAN INTERFACE TO THE FILE MANAGER

The MAX IV FORTRAN Interface to the MAX IV File Manager (FIFM) is a library of subroutines that can be called from FORTRAN programs to invoke File Manager Services. In this way, File Manager services can be accessed without the use of mechanisms such as in-line Assembly language coding.

For more information, refer to Section 6.5 of this manual and the MAX IV FORTRAN INTERFACE TO MAX IV FILE MANAGER, Library Reference Manual.

CHAPTER 5
SYSTEM PROCESSORS

An overview of each MAX IV system processor is presented in this chapter. System processors are discussed in terms of program development processors and file/data maintenance processors.

5.1 DEFINITION OF A SYSTEM PROCESSOR

The MAX IV System Processors are a collection of system utilities for program development and file/data maintenance and manipulation. The system processors are the tools with which applications are built and utility functions accomplished.

For detailed reference information on the use of each system processor, refer to the appropriate programmer's reference manual. The manual order numbers and titles of all system processor manuals are listed in the Preface of this manual. Refer to Figure 5-1 for an illustration of the system processors in the MAX IV OS.

5.2 TYPES OF SYSTEM PROCESSORS

System processors can be categorized as:

- o Program development processors
- o File/data maintenance processors
- o File Manager processors

Program development processors provide the tools needed for developing and maintaining software. Functions included are source code entry, object library creation and maintenance, linking, cataloging, and debugging. Overviews for processors in this category follow in this chapter.

File/data maintenance processors provide the system utilities needed for initializing and maintaining media, obtaining dumps, and copying disc/tape. Overviews for processors in this category follow in this chapter.

File Manager processors, also called utilities, provide listing, backup and restore, and disc preparation facilities within a system configured with the File Manager Subsystem. File Manager processors are overviewed in Chapter 4.

5.3 PROGRAM DEVELOPMENT SYSTEM PROCESSORS

MAX IV Program Development System Processors are used to create user programs. They include source and link editors and utilities for source/object file maintenance. Development processors are executed as overlays of Job Control.

5.3.1 SCREEN EDITOR

The MAX IV MODCOMP Screen Editor (MSED) is a display-oriented interactive text editor. When using MSED, the terminal screen acts as a window into the file that is being edited. Each screen is a set of 24 consecutive text lines of no more than 80 characters each.

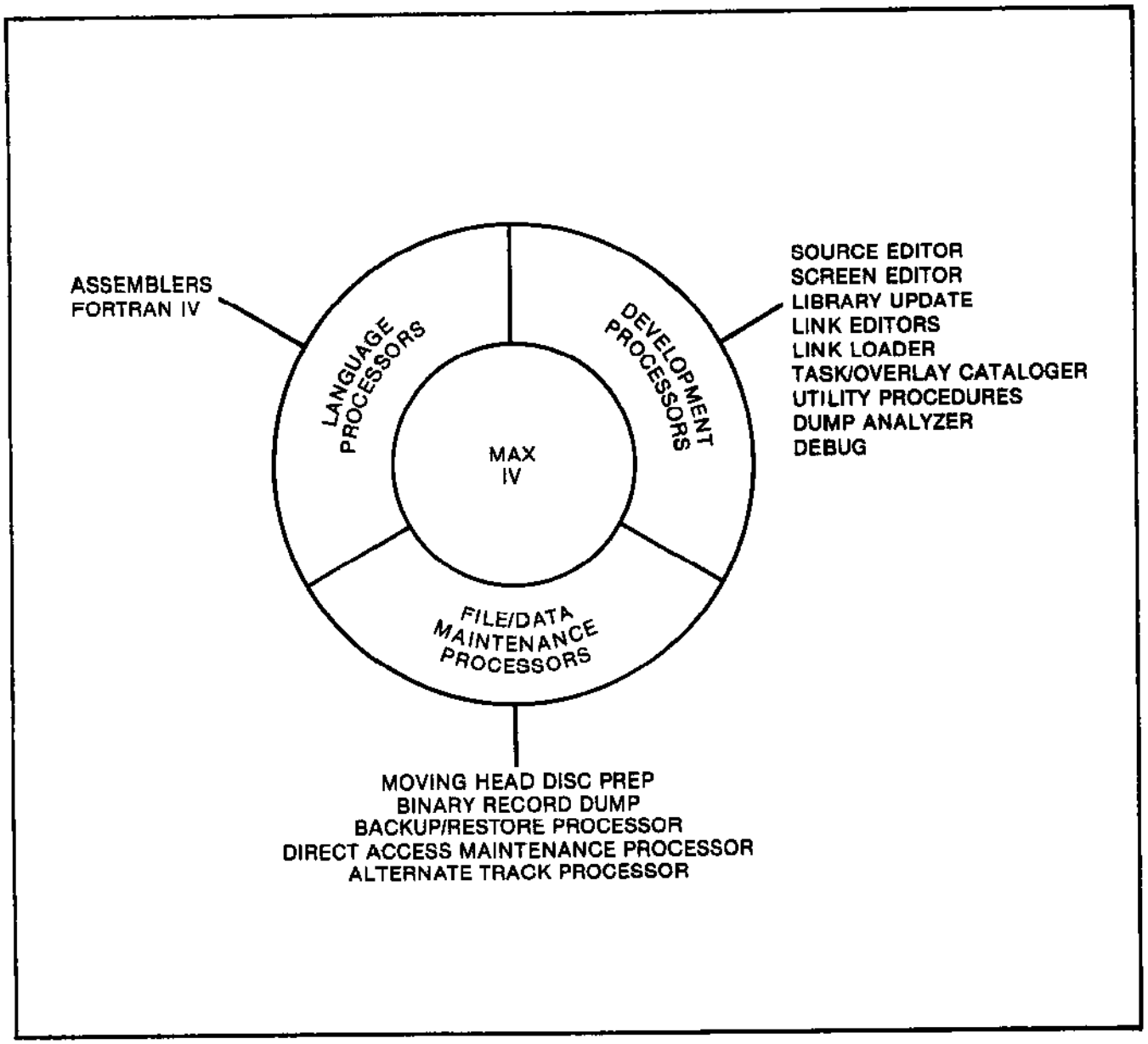


Figure 5-1. System Processors/Languages

Changes may be made to the text at any place in the current screen. The user positions the cursor to the appropriate character, word, or line, enters the proper command for the type of change required, and then enters the change itself. The change is completed by pressing the ESCAPE (ESC) key.

In addition to making changes to the text, the user can also copy a set of lines, search for a given pattern, read an additional file into the text, and write a set of lines from the text into a file on disc.

MSED is completely compatible with the MODCOMP Source Editor (SED). MSED works with the User Source Library (USL) and a subset of SED directives are executable directly from MSED. A user can enter, modify, catalog, delete, rename, and list source files and initialize a directory using MSED.

For detailed information, refer to the MAX IV MODCOMP SCREEN EDITOR, Programmer's Reference Manual, listed in the Preface.

5.3.2 SOURCE EDITOR

Source Editor (SED) is a sophisticated line editor for source program preparation and maintenance. SED supports both sequential and directory files. All MODCOMP assemblers and compilers can accept output from SED. For detailed information, refer to the MAX IV SOURCE EDITOR, Programmer's Reference Manual, listed in the Preface.

SED is used for source program:

- Creation
- Storage
- Updating
- Copying

5.3.2.1 Source Creation

The user of SED can input source code from any input device such as a card reader or a CRT. Source code can also be entered from sequential files or directory files, on disc partitions or on magnetic tape.

The integrity of the input data can be checked through sequence numbers and through checksum values in the case of input from magnetic media.

5.3.2.2 Source Storage

Source code can be stored sequentially on such media as magnetic tape or disc. In addition, source can be stored in a disc file that contains a directory. The user can catalog the source code file under a name and access it later by using that name.

By storing source files in a disc file with a directory, a library of source files known as a User Source Library (USL) can be created and maintained.

5.3.2.3 Source Updating

SED has two modes of editing:

- Sequential
- Random

In sequential mode, any changes to the source file must be made in sequence. SED processes the input record-by-record. Any additions, deletions, or changes must be made at the appropriate point.

In random mode, changes to the source file can be made in any order. A copy of the original source file is made on a scratch disc partition and the user can reference and modify records within the source in any order.

5.3.2.4 Source Copying

Source code can be copied from one media to another by using SED. For example, specific files from a USL or an entire USL can be copied to magnetic tape or to another disc file. When a complete USL is on tape, individual files can be accessed by name; however, files cannot be added or deleted as done on disc.

5.3.3 UTILITY PROCEDURES

The MAX IV Utility Procedures are a set of Job Control procedures used for program development and system initialization. While not an actual program (processor) in itself, the Utility Procedures are discussed in this section because, through Job Control, the procedures execute other system processors to accomplish program development functions. These procedures present a standard interface to the MAX IV OS. Functions performed through the utility procedures include program cataloging, various SED operations, and system generation.

For detailed information, refer to the MAX IV UTILITY PROCEDURES, Programmer's Reference Manual, listed in the Preface. In addition, two quick reference pocket guides exist for the experienced user of the MAX IV Utility Procedures.

A Job Control procedure is a sequence of Job Control directives, and directives for appropriate system processors, stored in a named file. The directives are executed in sequence when the name of the procedure is entered from Job Control. Procedures can be given parameters that are entered when the procedure is executed or that use default values.

5.3.4 LIBRARY UPDATE

Library Update (LIB) is a system processor used to build and maintain object libraries. LIB supports both sequential and directory libraries. Functions performed by LIB include such procedures as compressing object libraries, cataloging new object modules into libraries, and copying libraries. For detailed information, refer to the MAX IV LIBRARY UPDATE, Programmer's Reference Manual, listed in the Preface.

LIB directives can be categorized into the following:

- Directives that set the operational mode of subsequent directives.
- Directives that move object modules to and from libraries.
- Directives that perform utility functions.

LIB processes modules of binary object records in MODCOMP Standard Binary format only. The inputs to LIB are normally the unlinked object modules created by an assembler or high-level language compiler.

5.3.5 MAX III/IV LINK EDITOR

In a MAX IV environment, the MAX III/IV Link Editor is used during the Link-edit Phase of the system generation process to produce output compatible with the Stand-Alone Linking Loader (SAL). To avoid confusion, it is essential that the MAX IV user employ the MAX IV Link Editor to perform non-system generation link-edit functions. For detailed information, refer to the MAX III/IV LINK EDITOR, Programmer's Reference Manual, listed in the Preface.

5.3.6 MAX IV LINK EDITOR

The MAX IV Link Editor is a full-service editor that resolves incomplete binary object modules into a complete load module in a special editor format. The Link Editor permits the definition of a complex overlay structure and the association of common areas to private or global shared areas. For detailed information, refer to the MAX IV LINK EDITOR, Programmer's Reference Manual, listed in the Preface.

The MAX IV Link Editor is a processor capable of building object modules in an editor format from unedited object modules. It may be used for:

- o Converting unedited assembler/compiler object to an editor format.
- o Segmenting programs into overlay modules.
- o Allocating labeled common blocks in the addressing space.
- o Allocating labeled common blocks in extended addressing space.
- o Assigning attributes such as READ-ONLY, RELOCATABLE, and OPERAND-MAP to multiple location counters and common blocks.
- o Associating labeled common blocks with global or private shared regions.

The link-edit process is composed of two passes.

- o Link Edit Pass One - The primary object module is read and a symbol table is constructed of all internal definitions, external references, and common allocation. All external references are then resolved from either the specified input logical file, the optional User Library (UL) logical file, the System Library (LB) logical file, or from user-specified libraries. The validity of all data is checked during this pass.
- o Link Edit Pass Two - During the second pass, all references to external names are satisfied from the symbol table. A self-contained module is written on the Binary Output (BO) logical file.

5.3.7 LINK LOADER

The Link Loader is a system processor that loads a main object module into memory, together with any additional external object modules, and links the modules into an executable set of machine code. In the MAX IV environment, the Link Loader can only be executed as a Job Control overlay through the $EXECUTE Directive. For detailed information, refer to the MAX IV LINK LOADER, Programmer's Reference Manual, listed in the Preface.

5.3.8 TASK/OVERLAY CATALOGER

The Task/Overlay Cataloger (TOC) places fully-edited programs on disc in a directory library for subsequent action by the system loader. This process is known as cataloging. The cataloged object file is referred to as a load module. These load modules may be designated as tasks or as overlays of existing tasks. Programs may be cataloged in quick-load absolute format or may be made relocatable. For detailed information, refer to the MAX IV TASK/OVERLAY CATALOGER, Programmer's Reference Manual, listed in the Preface.

TOC catalogs object programs to be run under the MAX IV OS that are output from the MAX IV Link Editor and programs not requiring link-editing that are output from MODCOMP assemblers or compilers.

In addition, MAX IV programs have a number of associated characteristics and resources that dictate the environment in which the loaded program operates. TOC transparently allocates default values for these resources if they are not explicitly stated. TOC catalogs the resources together with the associated object program on a load-module file.

5.3.9 TYPICAL WORKFLOW OF PROGRAM DEVELOPMENT

A discussion of the typical workflow of program development follows. This discussion is intended to:

- Provide an over-all picture of the development process in terms of MODCOMP system processors.
- Show the inter-relationships among the key system processors discussed thus far.

This discussion follows the workflow illustrated in Figure 5-2.

The source code of a program is entered through the Source Editor or the Screen Editor. Depending on the computer language used, the source code is converted into object code by a compiler, by an assembler, or by a compiler and then an assembler.

If the program is entirely self-contained, the object module can be directly converted by the Task/Overlay Cataloger into a loadable task image called a load module. If the program references external routines, either explicitly by user reference or implicitly by the generated code, the link-edit phase of program development must obtain these routines from object module libraries and include them in the task image called the linked program. The Link Editor provides the linking facility and Library Update maintains the object module libraries. The Link Editor may need to scan several libraries, for example, a library of user-written routines as well as a library of language-support routines.

The linked program is then processed by TOC into a load module. The MAX IV Module Loader is used to load the task image into memory at execution.

5.3.10 DEBUG

Debug is a system processor used to debug assembly language programs. Debug provides debugging facilities in terms of machine instructions and memory addresses rather than in terms of symbolic labels and source line numbers. For detailed information, refer to the MAX IV DEBUG, Programmer's Reference Manual, listed in the Preface.

Debug is used more effectively in the interactive mode than in a batch mode. The programmer can enter commands and observe results in a sequence that is not predetermined. Debug may also be used in a batch mode through a fixed job stream.

Debug is always in one of two states: command state or execution state. In command state, Debug accepts and processes user commands. In execution state, the program being debugged is being executed. The transition from command state to execution state is accomplished through the use of the GOTO Directive. Transition from the execution state to the command state occurs when a breakpoint is found or when a violation or error occurs.

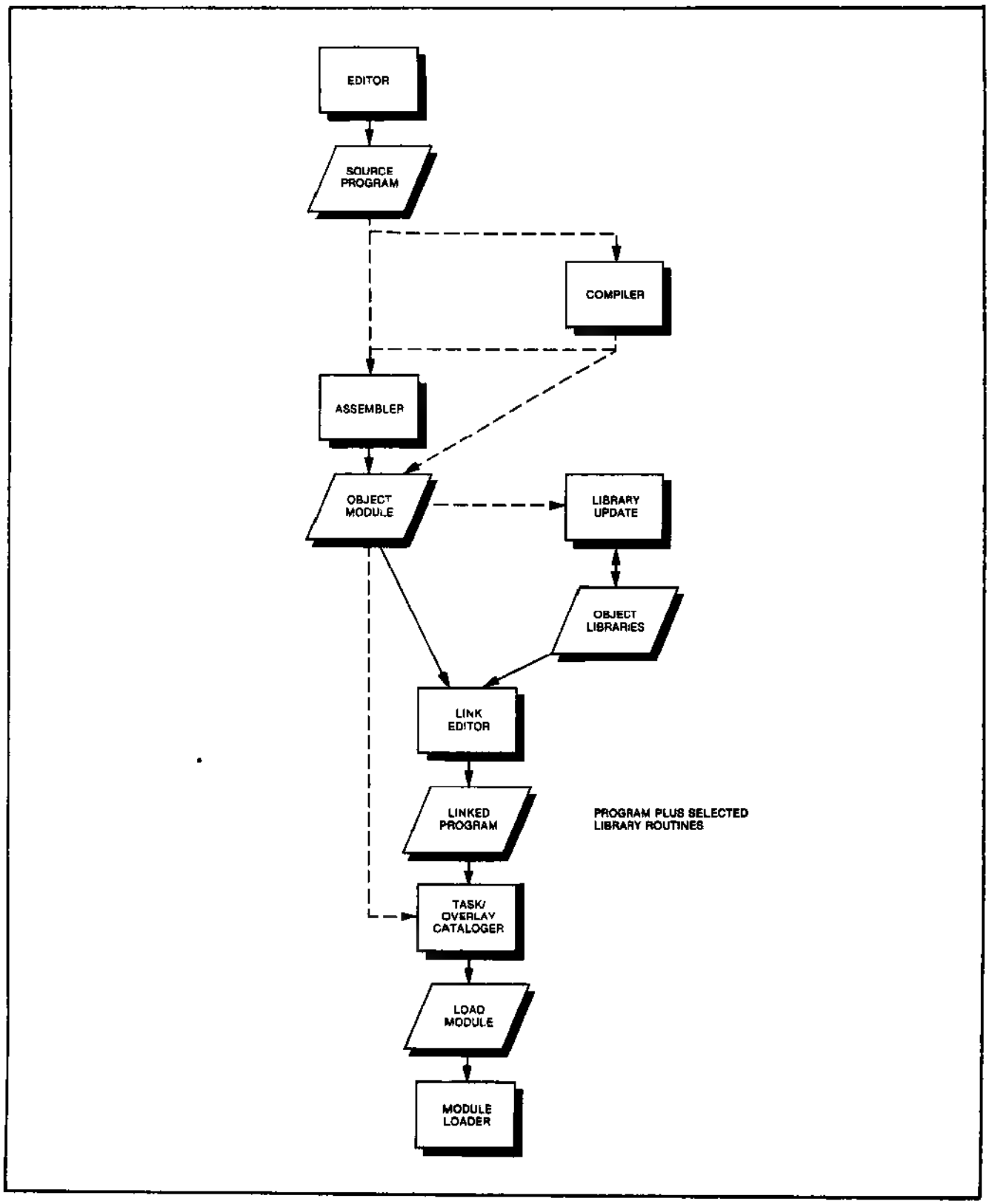


Figure 5-2. Program Development Workflow

5.3.11 DUMP ANALYZER

The Dump Analyzer, a SYSGENable option under the name ADUMP, is a set of software provided with the MAX IV OS that allows the entire contents of actual memory to be dumped to either disc or magnetic tape. In addition, this set of software provides a means of displaying the contents for analysis. This processor can help users in debugging their application tasks. For detailed information, refer to the MAX IV DUMP ANALYZER, Programmer's Reference Manual, listed in the Preface.

The main components of the Dump Analyzer are:

- o the Actual Memory Dump Routine
- o the Dump Analyzer Processor
- o the Online Copy Processor

The Actual Memory Dump routine is used in place of or in addition to the standard Stand-Alone Dump.

The Dump Analyzer Processor is an interactive program that reads the memory image on a disc partition, formats it, and outputs information to either a terminal or line printer. This processor has a HELP facility that explains the use of all directives.

When a dump is made to magnetic tape, it must be copied to disc before it can be analyzed. The Online Copy Processor performs the copy from tape to disc.

The Dump Analyzer includes the following features:

- o All actual memory is dumped. No information is lost even if it was not mapped at the time of the crash.
- o System downtime is reduced since the dump is written to disc or magnetic tape rather than to a line printer.
- o To facilitate use of the processor, directives follow the pattern of Operator Communications directives.
- o A dump image can be scanned interactively or printed on a line printer.
- o The Analyzer knows MAX IV equates. Equates can be obtained by a module of the Analyzer.

5.4 FILE/DATA MAINTENANCE SYSTEM PROCESSORS

MAX IV File/Data Maintenance System Processors are used for such utilities as initializing discs, obtaining dumps, and copying data. These processors also execute as overlays to Job Control.

5.4.1 DIRECT ACCESS MAINTENANCE PROCESSOR (DAMP)

The Direct Access Maintenance Processor (DAMP) is a system processor that provides the user with services in the following three areas:

- o Manipulating a FORTRAN direct-access directory on logical file RAD.

- Modifying the results of utility direct-access file operations with information from the direct-access directory.

- Performing utility operations on arbitrary sequential files. Since DAMP is targeted at direct-access file processing, operations on sequential files obey unusual rules.

For detailed information, refer to the MAX IV DIRECT ACCESS MAINTENANCE PROCESSOR, Programmer's Reference Manual, listed in the Preface.

With DAMP, the following functions can be performed:

- Create a new direct-access directory.

- Change information in, remove information from, or add information to a directory.

- Concatenate discrete directories into a single directory.

- List the contents of the current directory.

- Move directories from file to file.

- Display selected records from a file.

- Copy files to other files.

- Place duplicate values in each word of selected records of a file.

- Save files and restore them later.

- Establish a chain of directories for use by FORTRAN programs.

5.4.2 MOVING HEAD DISC PREPARATION

The Moving Head Disc Preparation (PREP) system processor initializes disc packs by writing in every sector with a special format. The data in each sector identifies the sector number and track number. A moving head disc cannot be used under the operating system until this formatting process has been performed. For detailed information, refer to the MAX IV MOVING HEAD DISC PREPARATION PROGRAM, Programmer's Reference Manual, listed in the Preface.

PREP is used when a disc pack is being prepared in an online mode; that is, under the control of the operating system. If disc preparation is done offline (not under OS control), the Moving Head Disc Initialization Package, part of the Stand-Alone Support Software, is used.

5.4.3 BINARY RECORD DUMP

The Binary Record Dump (BRD) system processor displays records as if written in MODCOMP Standard Binary Mode. BRD can also convert these records to a hexadecimal or decimal representation on a printing device. For detailed information, refer to the MAX IV BINARY RECORD DUMP, Programmer's Reference Manual, listed in the Preface.

In addition to the contents of each record, BRD provides the following information for each record:

- o Relative record number
- o File Position Index of the record
- o Size of the record in bytes
- o Hexadecimal value of the User File Table Status Word

5.4.4 BACKUP AND RESTORE

The Backup and Restore Processor (BRP) copies data from disc to magnetic tape or to disc, restores data from magnetic tape to disc, and copies BRP-formatted data from magnetic tape to magnetic tape. BRP provides the online means to create a backup copy of a volume and to restore that volume. BRP supports copy/restore operations on the volume level only. Other online processors provide the capability to manipulate individual files within a volume. For detailed information, refer to the MAX IV BACKUP AND RESTORE PROCESSOR, Programmer's Reference Manual, listed in the Preface.

BRP guides the user through the run procedure by way of an interactive dialog that solicits information from the user. Detailed help information is available at each prompt by entering a question mark.

To speed the copy process and to ensure backup reliability, BRP automatically verifies the copy as soon as it is completed. If a verification error occurs, BRP provides the cause of the error, the location of the error, and optionally, a dump of the conflicting blocks.

Read and write error messages are also explicit, giving the cause of the error and the User File Table contents. Advance, backspace, and rewind file commands provide the user with a means of positioning a tape.

BRP is an online processor in the sense that it runs under the operating system. It is not a real-time processor. That is, a volume should not be updated while it is being copied. BRP takes exclusive use of the output volume and, based on a user-specified option, exclusive use of the input volume.

5.4.5 ALTERNATE TRACK PROCESSOR

The Alternate Track Processor is a system processor that supports alternate tracking. It provides an online capability of updating the MODCOMP Error Map previously initialized by the MODCOMP Alternate Track Initializer, a part of the Stand-Alone Support Software. For detailed information, refer to the MAX IV ALTERNATE TRACK PROCESSOR, Programmer's Reference Manual, listed in the Preface.

After the initialization process, the Alternate Track Processor provides an on-line means to read the MODCOMP Error Map, display it, modify it, and write the corrected map back on the volume. The Processor, through a user-specified option, can also attempt to read data into its buffer from a track that has become defective and restore it to a newly assigned alternate track. It is the user's responsibility to determine the validity of the restored data. The processor limits the number of defective tracks that can be reassigned so as not to cause any alternate tracks to be assigned to the last cylinder or off the end of the disc.

CHAPTER 6
LIBRARIES

The MAX IV Operating System contains the following libraries as part of the standard software package:

- o FORTRAN Math Library
- o FORTRAN Run-Time Library
- o Utility Library
- o Executive Functions and Process I/O Library
- o FORTRAN Interface to File Manager
- o Magnetic Tape Labeled Volume Support Library
- o FORTRAN Interface to the Inter-Task Communication (ITC) Service

This chapter provides an overview of each library. Refer to Figure 6-1 for an illustration of the MAX IV Libraries.

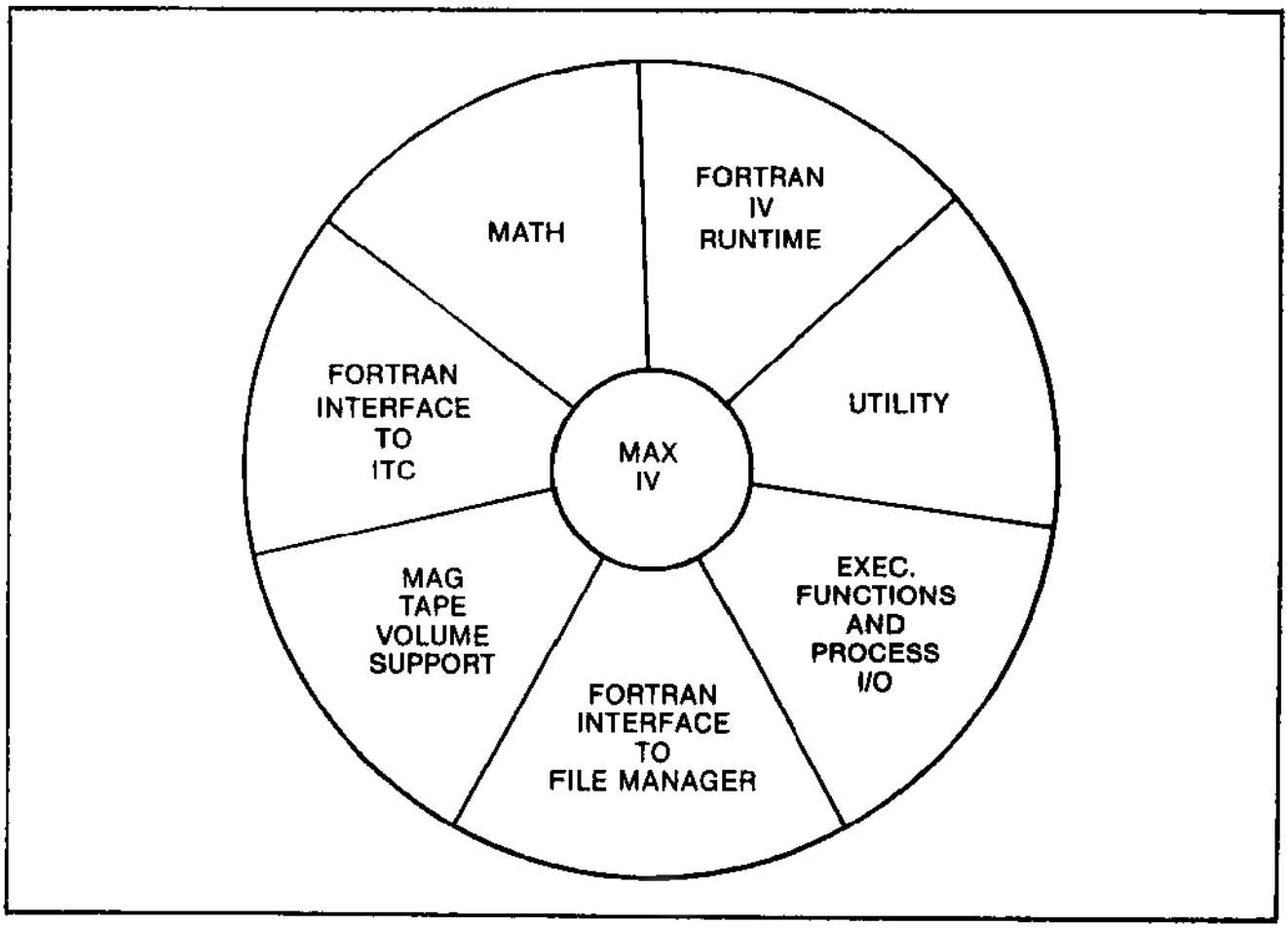


Figure 6-1. MAX IV Libraries

6.1 FORTRAN MATH LIBRARY

The FORTRAN Math Library is a collection of assembly language subroutines that enable optimum execution time under the MAX IV OS executing on a system that includes floating point hardware or software simulation of floating point. The subroutines of the library interface with the FORTRAN IV compiler and include all standard functions specified by the American Standard Association for FORTRAN.

The subroutines in the library are classified into:

- Subroutines that use MAX IV Executive Services
- Subroutines that are called by a FORTRAN program
- Support functions
- Special FORTRAN callable bit manipulation subroutines
- Subroutines called in the standard calling sequence
- Intrinsic functions

For detailed information, refer to the MAX IV MATH LIBRARY, Library Reference Manual, listed in the Preface.

6.2 FORTRAN RUN-TIME LIBRARY

The FORTRAN Run-Time Library is a collection of subroutines that provide an execution-time interface between a compiled FORTRAN program and the MAX IV Executive I/O services. The FORTRAN compiler generates links to the library that are satisfied at link-edit or link-load time.

The subroutines in the library can be grouped into six categories:

- Subroutines called by the compiled program
- Executive interface subroutines
- Format and list scanning subroutines
- Subroutines that transmit and convert values to/from user storage
- High-level ancillary subroutines
- Low-level ancillary subroutines

Called subroutines are the major control points of the library. The compiler generates direct linkages to these subroutines to perform specific actions requested in the source code.

Executive interface subroutines utilize REX calls to effect the actual data transfers and to link the proper physical devices.

The Format and List Scan subroutines determine the type and size of input/output conversion to be used during value transmission and perform other non-list interfacing chores.

The Value Transmission and Conversion subroutines encode/decode values and transmit them to/from the user storage area.

The High-Level Support subroutines convert from one radix to another, manipulate subfields within a character string, and generally provide service of a complex nature.

The Low-Level Support subroutines provide elementary services. These subroutines manipulate the stack and exit from subroutines, check the status of individual registers, perform multi-precision arithmetic, and so forth.

6.3 UTILITY LIBRARY

The Utility Library is a collection of subroutines that enable the use of any MAX IV REX service from a FORTRAN program. Each of the interfaces is capable of processing integer values of 16-bit or 32-bit length. The subroutines within the library are categorized as follows:

- o REX service interface subroutines. These subroutines are used to call the fundamental MAX IV REX services for file manipulation and task control.
- o Subroutines to aid MAX IV REX service interface. These subroutines provide support functions used by the standard REX service interface subroutines. For example, an important subroutine of this type is INTLEN, a subroutine that establishes the length of certain integer quantities used as arguments to other subroutines.
- o Subroutines related to MAX IV standard FORTRAN I/O. These subroutines are associated with FORTRAN I/O processing, such as searching for updated information in the RAD directory.
- o MAX IV auxiliary subroutines. These subroutines provide auxiliary functions such as obtaining elapsed time and console switch settings.
- o MAX III compatible REX service interfaces. These subroutines enable FORTRAN programs to be transported from the MAX III Operating System to the MAX IV Operating System.

For detailed information, refer to the MAX IV UTILITY LIBRARY, Library Reference Manual, listed in the Preface.

6.4 EXECUTIVE FUNCTIONS AND PROCESS I/O LIBRARY

The Executive Functions and Process I/O Library is a collection of subroutines that aid input/output interface. The subroutines in the library are categorized as follows:

- o Executive function subroutines. This category contains subroutines such as START by which a task can be activated after a designated time delay.
- o Process I/O subroutines. This category contains subroutines such as AISQW by which data can be read from a specified number of analog points in sequential order.
- o Bit Manipulation subroutines. This category contains subroutines by which functions such as INCLUSIVE OR and LOGICAL NOT are performed.
- o Time and Date subroutines. The subroutines in this category obtain the current system time and date.
- o Subroutines that aid I/O interface. The subroutines in this category provide such functions as building a MAX IV Process I/O User File Table from arguments provided.

For detailed information, refer to the MAX IV EXECUTIVE FUNCTIONS AND PROCESS I/O LIBRARY, Library Reference Manual, listed in the Preface.

6.5 FORTRAN INTERFACE TO THE FILE MANAGER

The MAX IV FORTRAN Interface to the MAX IV File Manager (FIFM) is a collection of FORTRAN callable subroutines that can invoke File Manager services. In this way, File Manager services can be accessed without the use of mechanisms such as in-line Assembly language coding. For detailed information, refer to the MAX IV FORTRAN INTERFACE TO MAX IV FILE MANAGER LIBRARY, Library Reference Manual, listed in the Preface.

The subroutines in FIFM fall into three categories:

- Service subroutines
- File Descriptor List (FDL) subroutines
- Error Processing Address (EPA) subroutines

Service subroutines interface directly to a File Manager service. They require the same operands as the File Manager. The Service subroutines include:

MOUNT
LABEL
RELABEL
FILEDESCRIBE
CREATE
REFILE
OPEN
EXPAND
CONTRACT
ENDFILE
CLOSE
DESTROY
DISMOUNT

FDL subroutines manipulate the File Descriptor Lists needed by the File Manager services. These subroutines also provide strategic pre-processing for EPA interfacing. The two FDL subroutines are:

- FDLINI - initializes the FDL.
- FDLGEN - translates file descriptors into Descriptor Identification Codes used by the File Manager.

The EPA subroutines translate the File Manager's Error Processing Address branching into a form compatible with the FORTRAN constraints and conventions.

For detailed information, refer to the MAX IV FORTRAN INTERFACE TO MAX IV FILE MANAGER LIBRARY, Library Reference Manual listed in the Preface.

6.6 MAGNETIC TAPE LABELED VOLUME SUPPORT LIBRARY

The Magnetic Tape Labeled Volume Support Library is a collection of subroutines that support the creation, modification, and reference of a subset of magnetic tape volume data structures as defined by ANSI standard.

The following tape label records are processed:

- o Beginning of volume (VOL1)
- o End of volume (EOV1)
- o Beginning of file (HDR1)
- o End of file (EOF1)

An Assembly language and FORTRAN user interface is provided. The following functions are supported:

- o OPEN - Associates file I/O assignments.
- o READBLOCK - Reads a file block.
- o WRITEBLOCK - Writes a file block.
- o REWINDFILE - Positions to beginning of file.
- o ADVANCEBLOCK - Bypasses one or more file blocks.
- o BACKSPACEBLOCK - Reverse one or more file blocks.
- o CLOSE - Disassociates file I/O assignments.

For detailed information, refer to the MAX IV MAGNETIC TAPE LABELED VOLUME SUPPORT LIBRARY, Library Reference Manual, listed in the Preface.

6.7 FORTRAN INTERFACE TO THE INTER-TASK COMMUNICATION (ITC) SERVICE

The FORTRAN Interface to the Inter-Task Communication (ITC) Service Library is a collection of FORTRAN callable subroutines that provide the user with a means of using ITC services to send messages from one task to another task. The subroutines contained in this library are categorized as:

- o Subroutines that accomplish the FORTRAN interface to the MAX IV ITC Service such as opening a virtual circuit.

- o Subroutines that provide ITC interface support functions such as setting/resetting the Error Reporting option of ITC.

For detailed information, refer to the MAX IV FORTRAN INTERFACE TO INTERTASK LIBRARY, Library Reference Manual, listed in the Preface.

CHAPTER 7
LANGUAGES

The MAX IV Operating System contains the following language support as part of the standard package:

- o Assembly
- o Assembly Cross-Reference Program
- o FORTRAN IV
- o FORTRAN 77

7.1 ASSEMBLERS

The MODCOMP Assemblers are language processors that define the conventions of the assembly language as it pertains to MODCOMP computers. The MODCOMP Assemblers include many features that assist in the programming of MODCOMP computers. Each of the following features is described in detail in the MODCOMP ASSEMBLERS, Language Reference Manual, listed in the Preface.

The MODCOMP Assemblers contain the following features:

- o Mnemonic operation codes. Each machine instruction for MODCOMP computers has a mnemonic operation code assigned to it for use in the assembly language.

- o Directives. Directives are mnemonics recognized by the Assemblers that cause specific functions to be performed during the assembly of a source program. An extensive set of directives exist to:

  - Express constants.
  - Allocate storage.
  - Perform inter-program communication.
  - Format listed output.

- o Free Field Format. Specific columns are not required for input to the Assemblers.

- o Symbolic Addressing. Symbolic addressing allows the use of symbols rather than absolute addressing for referencing memory locations, general registers, bits, or any other desired field.

- o Assembly of data. Machine language values may be represented in decimal, hexadecimal, and character strings.

- o Relocatable assembly. MODCOMP Assemblers produce an object format that may be loaded by the Linking Loader at various locations of memory. An absolute mode of assembly is also available.

- o Linking program sectors. Programs may be written in separately assembled sections and linked together at load time by the Linking Loader. References may be made between sections by the INT and EXT directives.

- Extension of instruction set. The Assemblers may be extended to include custom hardware macros through use of the DFF and DFC directives. This extension does not involve any modification to the Assemblers.

- Object listing. The MODCOMP Assemblers produce an object listing consisting of the source statements and the object program produced by the assembly.

- Error diagnostics. As each source program is being assembled, any errors detected are noted by the Assembler on the object listing. Major errors, such as undefined operation codes and symbols, are output during the first pass of assembly. This can save considerable debugging time.

- Macro definition. Any allowable sequence of code may be utilized to define a macro including nested macros, that is, calls to other macros. Defined macros take precedence over existing mnemonic operation codes.

- Macro utilization. Defined macros are called by placing the name of the macro in the operation field of the source line. Any legal name may be utilized as the name of the macro. In addition, up to sixteen symbolic arguments may be passed to the macro by placing them on the calling line.

- Conditional assembly directives. A full set of directives that test logical and environmental conditions of symbolic expressions permits the user to assemble or to bypass selected sequences of source code. In addition, an unconditional GOTO Directive permits the user to further extend the utility of the conditional statements.

- INSERT Directive. Sequences of code or blocks of macros may be inserted into the macro language at any point through the use of this directive.

- COMMON. The Assemblers permit the definition, initialization, and referencing of COMMON in a manner that permits communication between the assembly language program and FORTRAN and/or other assembly language programs.

- Fixed/Floating Point Constants. Floating point constants or scaled binary constants may be created with the use of this feature.

7.2 ASSEMBLER CROSS-REFERENCE PROGRAM

The Assembler Cross-Reference Program (XREF) produces a cross-reference listing of symbols used in a source program written in MODCOMP assembly language. This listing is very helpful to the maintenance or development programmer by providing quick identification of all references to a symbol, which greatly simplifies the tasks of debugging and enhancing.

Source input to XREF can be any MODCOMP assembler language file. XREF accepts source programs in standard ASCII format and MODCOMP compressed ASCII format.

The output from XREF consists of the first TTL line found (this identifies the listing) followed by symbols in alphanumerical order with their references. In addition, XREF provides a number of options and commands for tailoring the cross-reference listings to suit the requirements of the particular programming task.

For detailed information, refer to the MAX IV ASSEMBLER CROSS-REFERENCE PROGRAM, Programmer's Reference Manual, listed in the Preface.

7.3 FORTRAN IV

The FORTRAN IV compiler (FR5) complies with the specifications outlined by ANSI FORTRAN X3.9-1966. The FR5 compiler includes the following modifications and enhancements to facilitate the writing and debugging of application programs:

- Identifiers may be of any length but only the first six characters are used by the compiler.
- Expressions may contain variables of mixed mode.
- Subscripts may be made up of any expressions.
- A main program may be named.
- A variable number of arguments may be passed to a subroutine.
- Hexadecimal constants are allowed in DATA statements.
- Inline coding of MODCOMP assembly language instructions is allowed.
- Array initialization in a DATA statement is allowed.

In addition, the MODCOMP FORTRAN Debug Package is provided. This package is a comprehensive debugging tool that is selected through a compile time option. It requires no modification of the FORTRAN source program to be debugged.

A set of interactive directives allow pauses to be placed in the program at specified lines and various types of traces. In addition, system violations are detected and information is displayed indicating the chain of events leading to the violation.

For detailed information, refer to the MAX IV FORTRAN IV, Language Reference Manual, listed in the Preface.

CHAPTER 8
OPTIONAL SUPPORT SOFTWARE

The MAX IV Operating System interfaces with optional support software in each of the following categories:

- o Redundant software system - LIFEGUARD
- o Communications/networking software
- o File/data maintenance software
- o Language Support

This chapter provides an overview of all optional support software in these categories. Refer to Figure 8-1 for an illustration of the MAX IV optional support software.

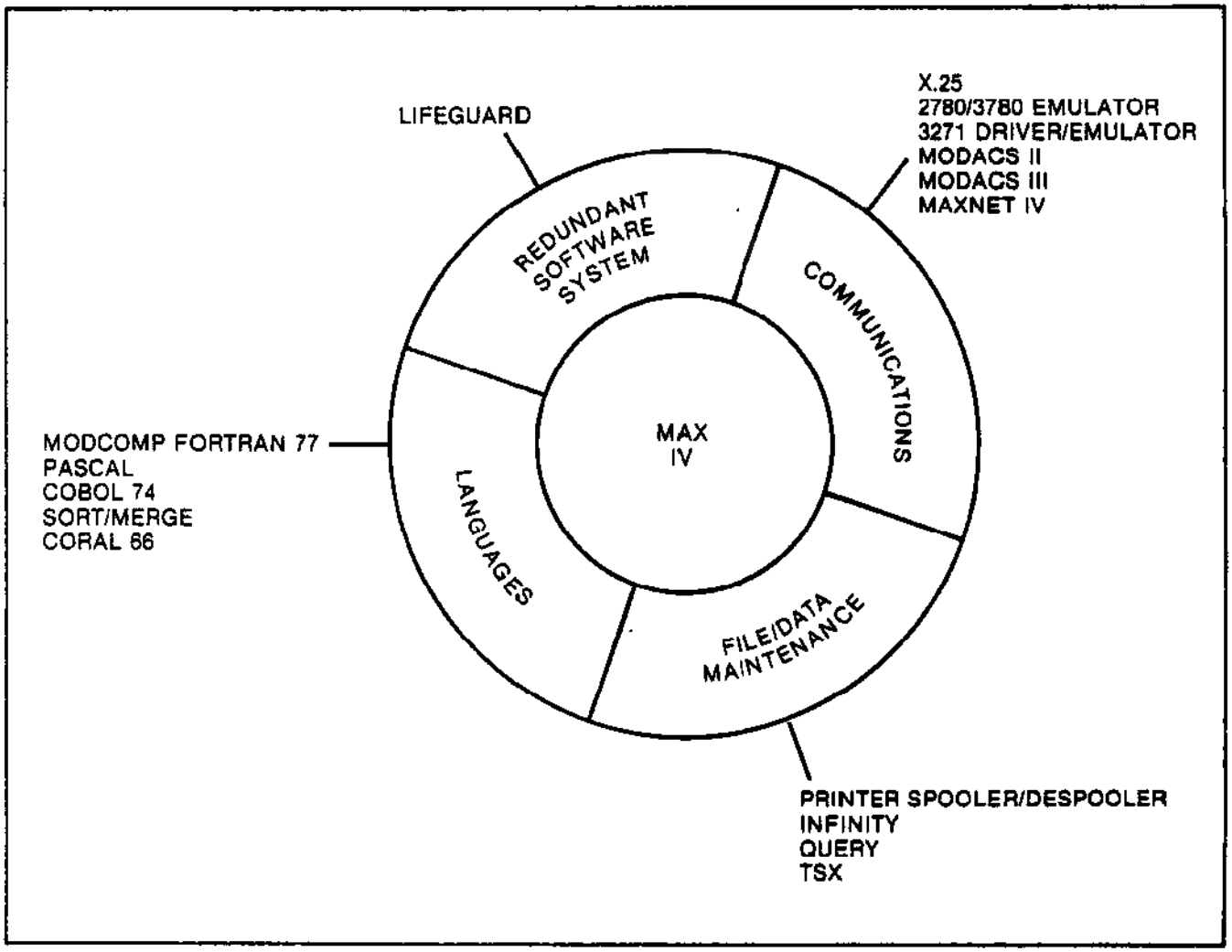


Figure 8-1. MAX IV Optional Support Software

8.1 REDUNDANT SOFTWARE SYSTEM - LIFEGUARD

LIFEGUARD is a redundant software system, operating in a MAX IV environment, that supports the operation of two MODCOMP CLASSIC CPUs, in a primary/backup configuration. In the event of system failure, switchover to the other CPU can be accomplished to process a given set of tasks. LIFEGUARD executes as a resident task on both CPUs. The Standard Communications Link (4828) handles the communication between the two processors.

LIFEGUARD is composed of the following three programs:

- o Stall Detector Program
- o LIFEGUARD Executive Program
- o Dual Input/Output Symbiont

The Stall Detector Program detects failures in the primary CPU. This is accomplished by monitoring the primary CPU for the condition in which a task is exclusively using the processor for a given period of time. When a stall is detected, a switchover condition is assumed and processing is initiated on the backup CPU.

The LIFEGUARD Executive Program, implemented as a symbiont, provides:

- o CPU control
- o Interprocessor link
- o Fault detection and response
- o Switchover and switchback processing
- o Peripheral switching

The Dual Input/Output Symbiont (DIOS) supports switchover and switchback processing by providing a method for performing redundant REX services in a dual CPU environment. I/O requests for a device associated with either CPU can be satisfied with a backup CPU device. This feature is optional.

Refer to Figure 8-2 for an illustration of the relationships between LIFEGUARD and the primary and backup CPUs. For detailed information, refer to the MAX IV LIFEGUARD, Programmer's Reference and System Design Manuals, listed in the Preface.

8.2 COMMUNICATIONS/NETWORKING SOFTWARE

MAX IV supports a wide variety of software for applications in distributed and non-distributed communication networks. The communication software products supported by MAX IV are:

- o X.25
- o 2780/3780 Emulator
- o 3271 Driver/Emulator
- o MODACS II Process I/O Subsystem
- o MODACS III Process I/O Subsystem
- o MAXNET IV Distributed Processing Extension

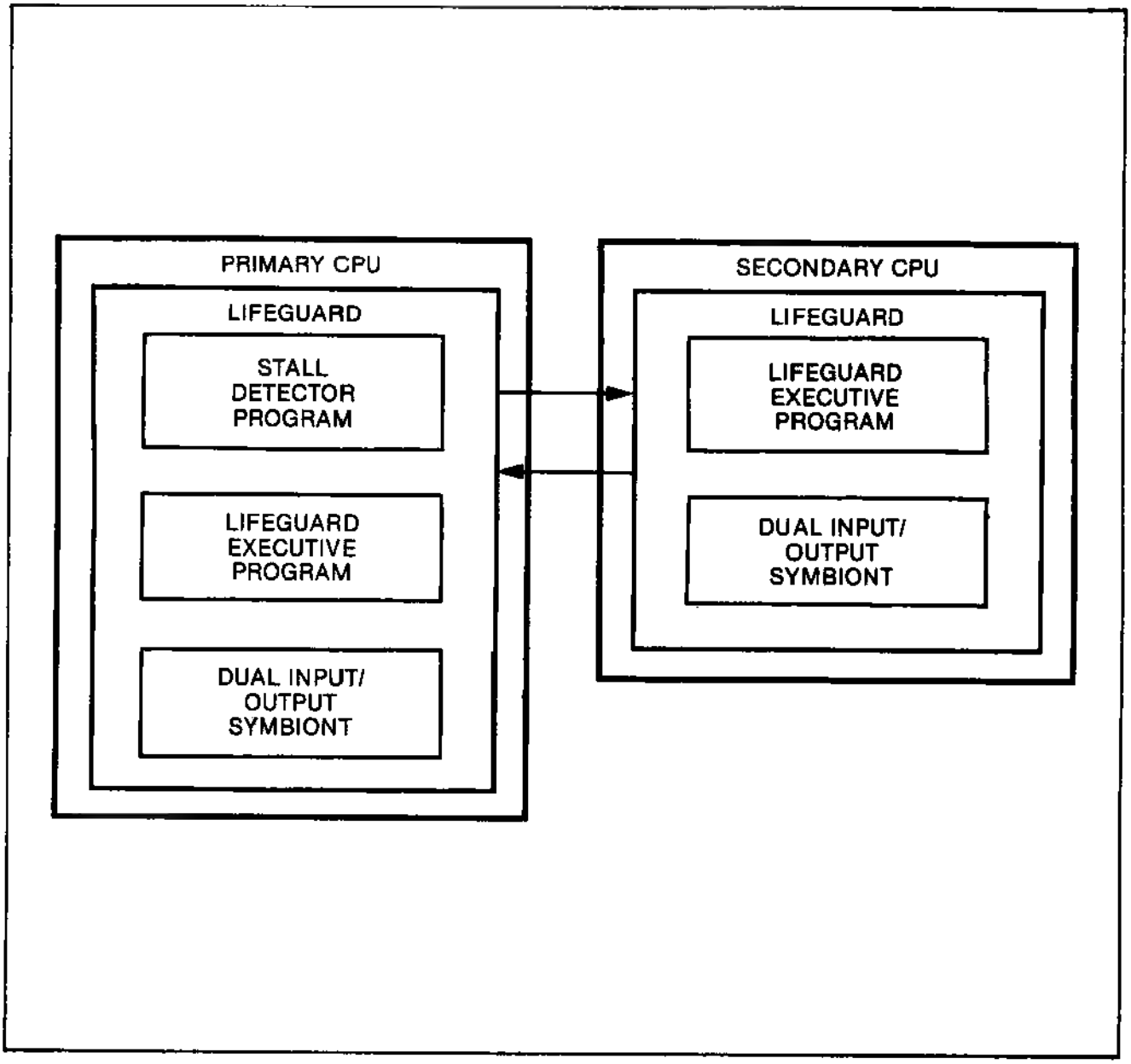


Figure 8-2. LIFEGUARD

8.2.1 X.25 COMMUNICATIONS SOFTWARE

The MODCOMP X.25 communications software consists of the following:

- The X.25 task that supports the Level 2 and 3 functions defined in the CCITT Recommendation X.25, Geneva, 1977. In addition to its applicability to private networks, it supports connections into the following public packet-switched networks:
  - Euronet (European Economic Community)
  - Transpac (France)
  - PSS (UK)

- The Call Control task that provides a simple interface for setting up calls across a network through X.25.
- The Event Reporting Task that outputs information about events or errors in the X.25 task. This task is optional.
- Operator communication directives that control the X.25 link and output statistics and other information about the X.25 task.

Communication among the user, the Call Control task, and the X.25 task are handled by the MAX IV Inter-task Communications Service. The relationship of the various parts of the MODCOMP X.25 software is illustrated in Figure 8-3. For detailed information, refer to the MODCOMP X.25, Communications Reference Manual, listed in the Preface.

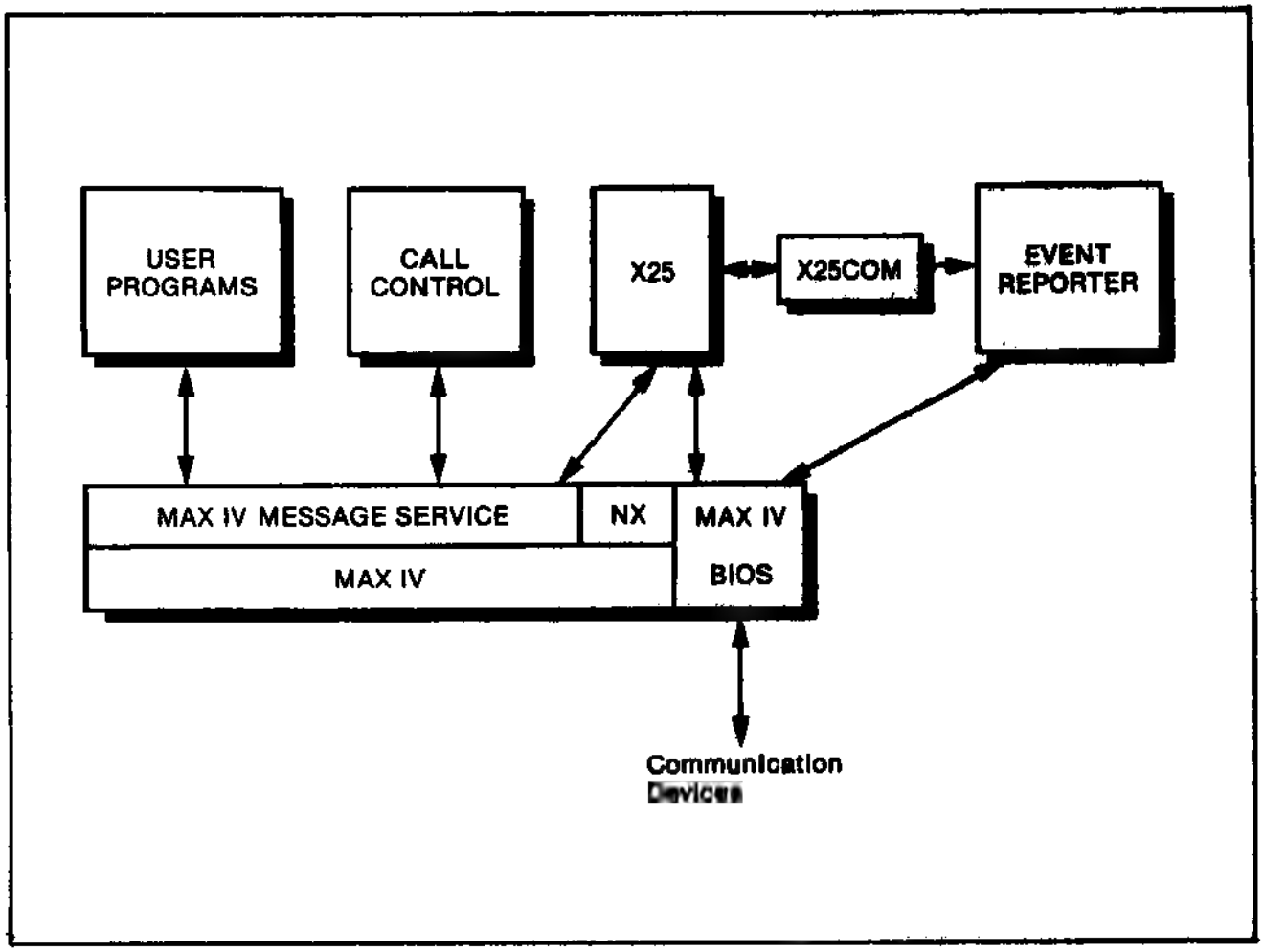


Figure 8-3. MODCOMP X.25 Software

8.2.2 2780/3780 TERMINAL EMULATOR

The 2780/3780 Terminal Emulator provides the means for a MODCOMP computer to communicate with a remote computer that supports 2780 or 3780 Remote Job Entry (RJE) terminals.

Both the 2780 and the 3780 RJE terminals allow large volumes of card data to be transmitted over communications lines, resulting in punched card or printed output. Communication is accomplished using Binary Synchronous Communication Procedures over leased, privately owned, or switching networks.

The MODCOMP 2780/3780 Emulators provide MODCOMP system users the capability to access the RJE services of a large IBM or IBM-compatible host computer. The Emulators perform such functions as:

- o Submitting source programs for assembly or compilation.
- o Submitting data for remote processing.
- o Submitting object programs for execution.
- o Receiving and listing the printed output of submitted jobs.
- o Receiving 80-character source or object records in the form of card punch records.

These functions are accomplished by emulating most of the functions of a 2780 or 3780 terminal through emulation programs executed under the MAX IV Operating System. In addition, operational advantages are offered by the Emulators. For example, on an actual 2780/3780 terminal, input is restricted to the card reader and output to the printer. In an emulation mode, input may be assigned to card reader, disc, or magnetic tape and output may be assigned to printer, disc, tape, or output spooler.

For detailed information, refer to the MAX IV 2780/3780 EMULATOR, Programmer's Reference and System Design Manuals, listed in the Preface. Refer to Figure 8-4 for an illustration of the flow of emulator operation.

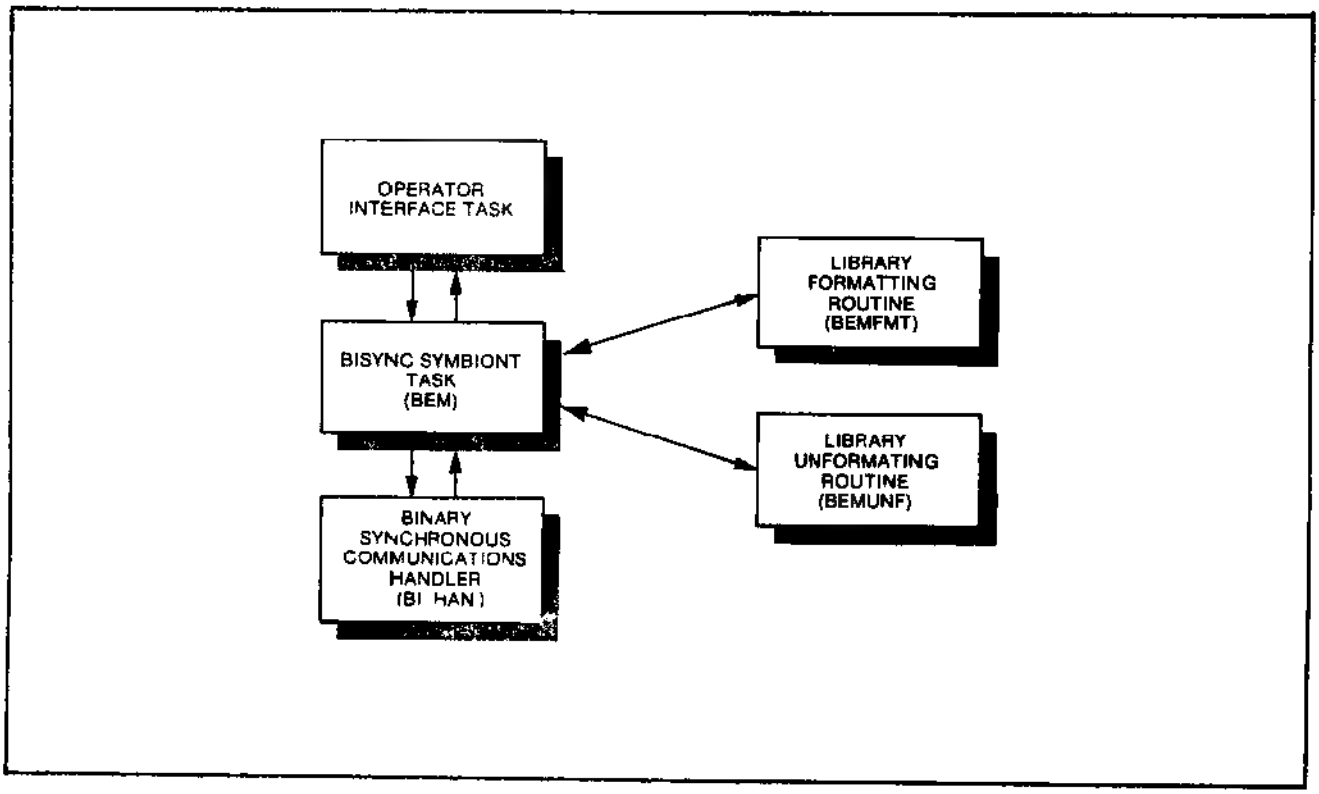


Figure 8-4. Flow of 2780/3780 Emulator Operation

8.2.3 3271 DRIVER/EMULATOR

The 3271 Driver/Emulator is a software package that enables a CLASSIC series computer to function as both:

- o A 3271 Control Unit (CU) Emulator
- o A 3271 Transmission Control Unit (TCU) Driver

When operating as a 3271 Control Unit Emulator, the software performs all the functions required to communicate with attached devices and pass data to and from the host processor through dedicated point-to-point or multipoint channels. In this capacity, it allows the attachment of up to 32 polled CRT terminals or printers that operate in a 3270 environment.

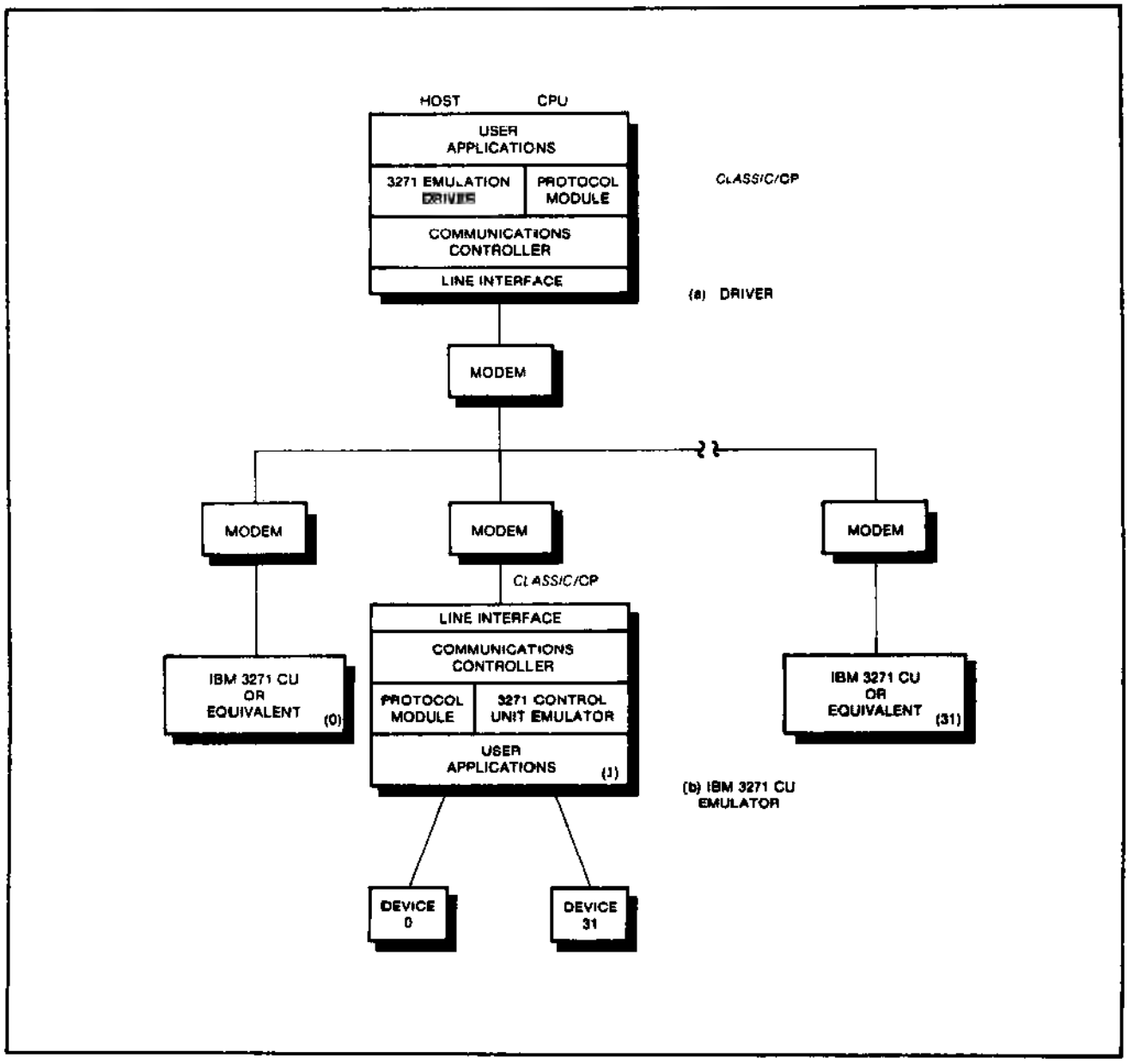


Figure 8-5. 3271 Driver/Emulator

When operating as a 3271 Transmission Control Unit Driver, the software performs the functions of a host processor controlling a multipoint channel with a network of 3271-compatible Control Units attached.

The 3271 Driver/Emulator package operates under the MAX IV OS. This enables the package to share computer resources and perform the emulation tasks concurrently with other real-time or batch tasks.

For detailed information, refer to the MAX IV 3271 DRIVER/EMULATOR, Programmer's Reference and System Design Manuals, listed in the Preface. Refer to Figure 8-5 for an illustration of the configuration of the system with both a CU Emulator and a TCU Driver.

8.2.4 MODACS II PROCESS I/O SUBSYSTEM

The Modular Data Acquisition and Control Subsystem (MODACS II) is a 16-bit microprocessor-based data acquisition and control system. The MODACS II interfaces directly with standard MODCOMP communications subsystems and provides alternatives to the present MODCOMP MODACS III Subsystem.

The typical MODACS II system is located in a remote location. The MODACS II software supports the MODACS II hardware system and allows existing applications packages to communicate with MODACS II as if it were a local MODACS III hardware system, through minor modifications. The MODACS II software system is illustrated in Figure 8-6

MODACS II provides these standard features:

- Continual data acquisition
- Response when polled by a host computer only
- Interpretation of commands sent by terminal or communications port from the host CPU
- Performance of requested functions
- Interpretation of significant changes signaled by the controller

The MODACS II Host Software System consists of the following modules:

- Process I/O Support Library (MODACL)
- REX, #21 Service (MDH)
- MODACS II Handler Symbiont (RIOHAN)
- Asynchronous DI/DMI Handler (AS.RIO)
- Installation Procedures
- MODACS II/III Macro File

MODACS II can be used with the CLASSIC II series. Only asynchronous channels in the 1907 and 1908 communications subsystems are supported by the host software. Both the RS-232 and the current loop interfaces are supported. The 1907 and 1908 controllers are supported in both DI (Data Interrupt) and DMI (Direct Memory Interface) modes.

MODACS II is capable of multi-drop operations over dedicated wire or dedicated common carrier facilities (using a low cost modem). Dial-up lines are not supported. Baud rates from 300 bps to 19.2K bps are supported.

MODACS II can be configured in a Single-Drop or Multi-Drop Network. Its five major components are:

- o Host software
- o Microprocessor-based controller
- o Firmware
- o Power supply
- o Card file

MODACS II and III are released as part of the same software subsystem and share certain modules. Figure 8-6 illustrates the relationship between the MODACS II handler and the MODACS III handler. The application program (shown at the top of Figure 8-6) issues a MODACS command either through a FORTRAN call or the REX #21 service routine, MDH. The REX #21 service determines if the call is directed to MODACS II or MODACS III, then routes the command to the appropriate handler.

For additional information, refer to the MODACS II PROCESS I/O SUBSYSTEM, Programmer's Reference Manual listed in the Preface.

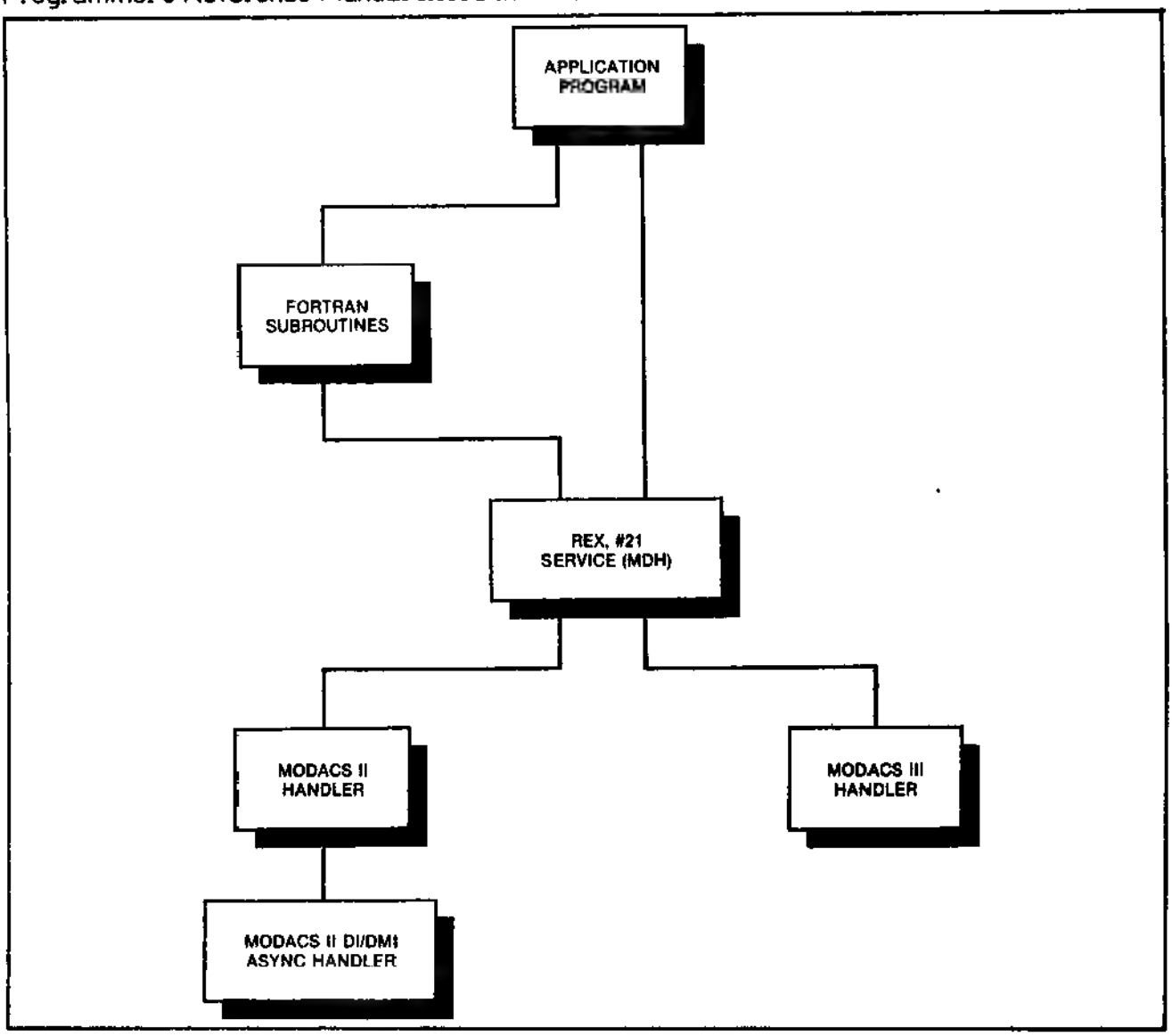


Figure 8-6. MODACS II and MODACS III Handler Relationship

8.2.5 MODACS III PROCESS I/O SUBSYSTEM

The MODACS III Process I/O Subsystem provides the software for communicating with the MODACS hardware subsystems. The MODACS III PIO Subsystem includes:

- A Process I/O Handler
- A support library
- The Common Alarm Events Monitoring task
- The Multifunction Counter Events Monitoring task
- A macrofile for configuring MODACS hardware subsystems
- A macrofile for assembling the Handler, Support Library, and Events Monitorings tasks (source code form only)
- Job Control installation procedures for installing a MODACS system

The Handler operates in the MAX IV Operating System environment independently of the Basic Input/Output System (BIOS). The Handler utilizes the MODACS III Logical Input/Output System. The Handler performs I/O directly to/from the calling task thereby providing low overhead and fast response.

For detailed information, refer to the MODACS III PROCESS I/O SUBSYSTEM, Programmer's Reference Manual, listed in the Preface.

8.2.6 MAXNET IV

MAXNET IV is a MAX IV OS extension that provides distributed processing capabilities. A typical MAXNET IV configuration is illustrated in Figure 8-7. Satellite computers interface to physical sensors as user programs in the satellites prepare the data to be sent across communications links to the host computer. At the host, user programs perform data reduction for storage on the files located there. In addition, other peripherals located at the host are utilized and accessible across the link.

The MAXNET IV capabilities are provided by a number of modules that may be included in the system at system generation. A system can be customly tailored by including only those modules required. The modules are:

- A link task that interfaces to the logical I/O system and allows device-independent I/O transfers over up to 16 computer-to-computer links.
- An optional loader symbiont that enables the loading of tasks in a satellite computer from a host system.
- An optional simultaneous output task that broadcasts data to two devices.
- An optional loader task that provides remote fill capability.
- An optional memory device interface that the capability to transfer data to and from memory buffers defined at system generation.

8.3 FILE/DATA MAINTENANCE SOFTWARE

The MAX IV OS supports optional file/data maintenance and manipulation software. These software packages include:

- o The Printer Spooler/Despooler Subsystem
- o INFINITY - a data base management system
- o Interactive Inquiry and Report Writer (QUERY) - a data base inquiry program for use with INFINITY
- o Time Sharing Executive/Transaction Processor (TSX)

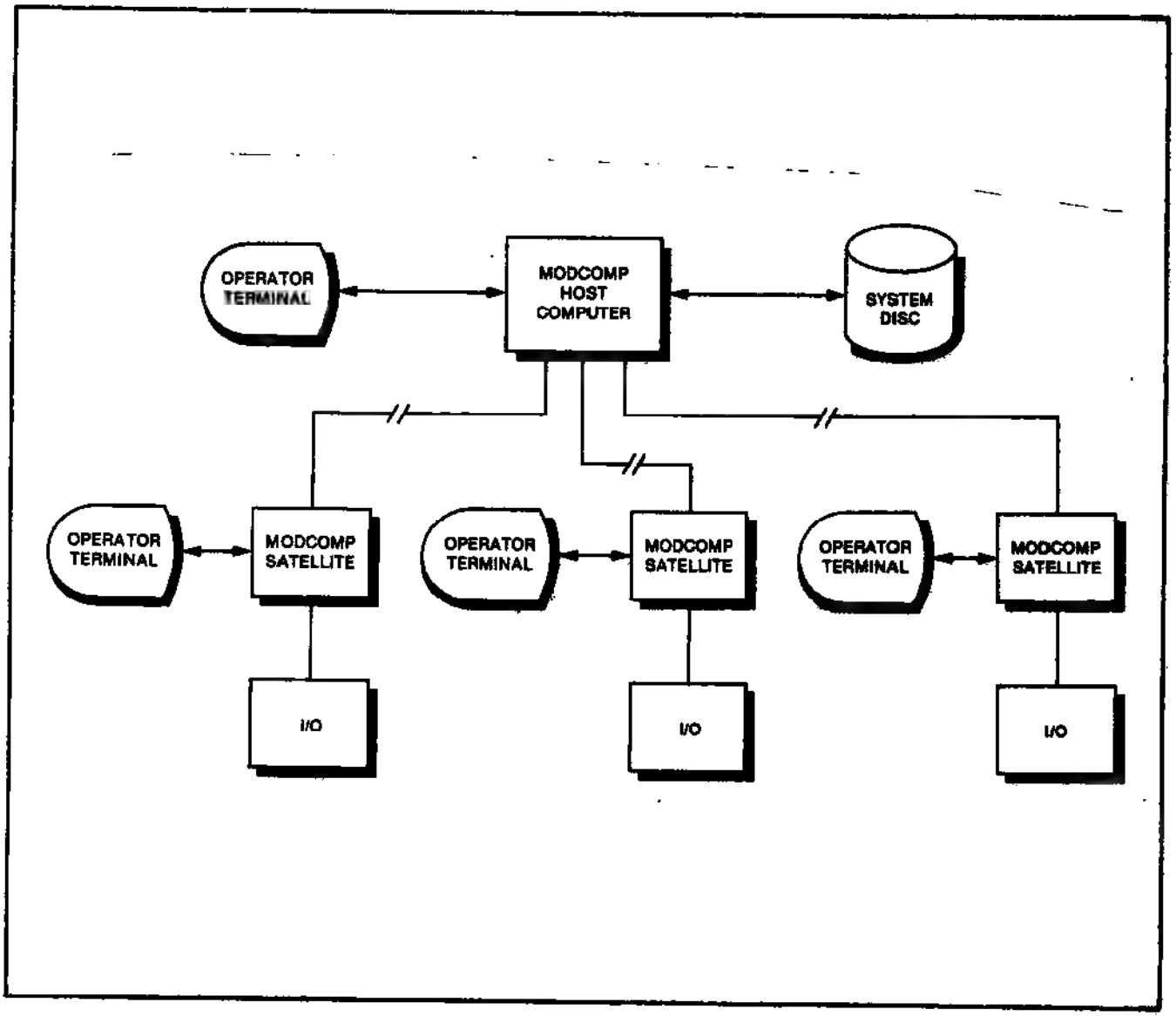


Figure 8-7. Typical MAXNET IV Configuration

8.3.1 PRINTER SPOOLER/DESPOOLER SUBSYSTEM

The MAX IV Printer Spooler/Despooler Subsystem controls the printing of multiple listings in a large multi-user environment. Used as an alternative to the standard MAX IV Spooler (S) task, the subsystem provides additional capabilities to better control the despooling of printed output.

The subsystem provides the capabilities to:

- Identify print file characteristics
- Control the printing process
- Manage the print queue
- Handle physical printer devices

The subsystem contains three symbionts that, along with several OC directives and two overlays, perform the following functions:

- The Spooler Symbiont accepts write requests, creates files using the File Manager, and simulates the line printer end-of-form detection.

- The Queue Manager Symbiont receives write requests from the Spooler Symbiont, maintains a list of files contained in the print queue, and accepts operator input to change or delete entries in the queue.

- The Despooler Symbiont drives the printers. It receives requests from the Queue Manager Symbiont that contain the name of the file to be printed and the destination print device.

For detailed information, refer to the MAX IV PRINTER SPOOLER/DESPOOLER SUBSYSTEM, Programmer's Reference Manual, listed in the Preface.

8.3.2 INFINITY

INFINITY is a general-purpose data base management system for the CLASSIC computers. INFINITY supports multi-program, multiprocessor access to data base files through the MAX IV OS logical I/O structure. INFINITY operates as a resident non-MAP 0 symbiont task processing queued user requests. Nonresident system data base processors are included for general-purpose data entry, retrieval, and file maintenance.

Schema-driven files are created using a Data Description Language (DDL) that describe data record items, access structures, and other attributes. A set of callable routines provide a convenient Data Manipulation Language (DML) interface with REX I/O functions. INFINITY is language-independent, supporting FORTRAN IV, PASCAL, COBOL 74, and assembly language interfaces.

Since INFINITY can also use the MODCOMP MAXNET IV, INFINITY data can be accessed directly from satellite systems executing different applications. Multiple INFINITY systems may be configured to provide distributed data base systems with transparent multi-user access from any point in the network.

The following features are provided by INFINITY:

- Schema Data Definition Language (DDL) for describing the data base.

- o Language independent Data Manipulation Language (DML) subroutines that support data management functions as extensions to the host language.
- o Record-level and file-level locking for data security and integrity.
- o Two levels of space management: free space and extents.
- o A record of all operations affecting the structure or data content of each file can be generated.
- o Files organized up to eight methods simultaneously.
- o Multi-volume files.
- o Transparent file segmentation.
- o Automatic and manual file expansion.
- o Distributed system architecture.

Additionally, this software package can be combined with QUERY, Interactive Inquiry and Report Writer, and TSX, Time Sharing Executive/Transaction Processor, for interactive processing.

INFINITY provides a transparent, schema-driven data record organization system for multi-keyed data files as depicted in Figure 8-8. Records may be retrieved by specifying partial or complete key data values. Direct access dynamic index structures transparent to the user provide exceptionally rapid access to data records. Keys may be specified to locate a single record or to locate the first or last occurrence of the data item. Sequential access in ascending or descending key order can locate additional records until the selection criteria is exhausted. This structure is ideal for multi-user interactive environments since no sorting or scanning techniques are used. Index structures dynamically expand or contract as data records either are added or deleted from the database.

For detailed information, refer to the INFINITY, Data Base Management System, Data Management Manual, listed in the Preface.

8.3.3 QUERY

The Interactive Inquiry and Report Writer (QUERY) is a data base inquiry/report program that accesses the information in an INFINITY data base. Both automatic and manual report writer features are provided as well as a simple QUERY editor for ad hoc report requests, complex reports, and easy composition and storage of the QUERY language request.

QUERY uses a data dictionary structure that permits the use of meaningful data item names in requests and that controls report format and security. Actual information spread across multiple data base files is automatically related by QUERY. The user of QUERY need not be aware of the actual file structures.

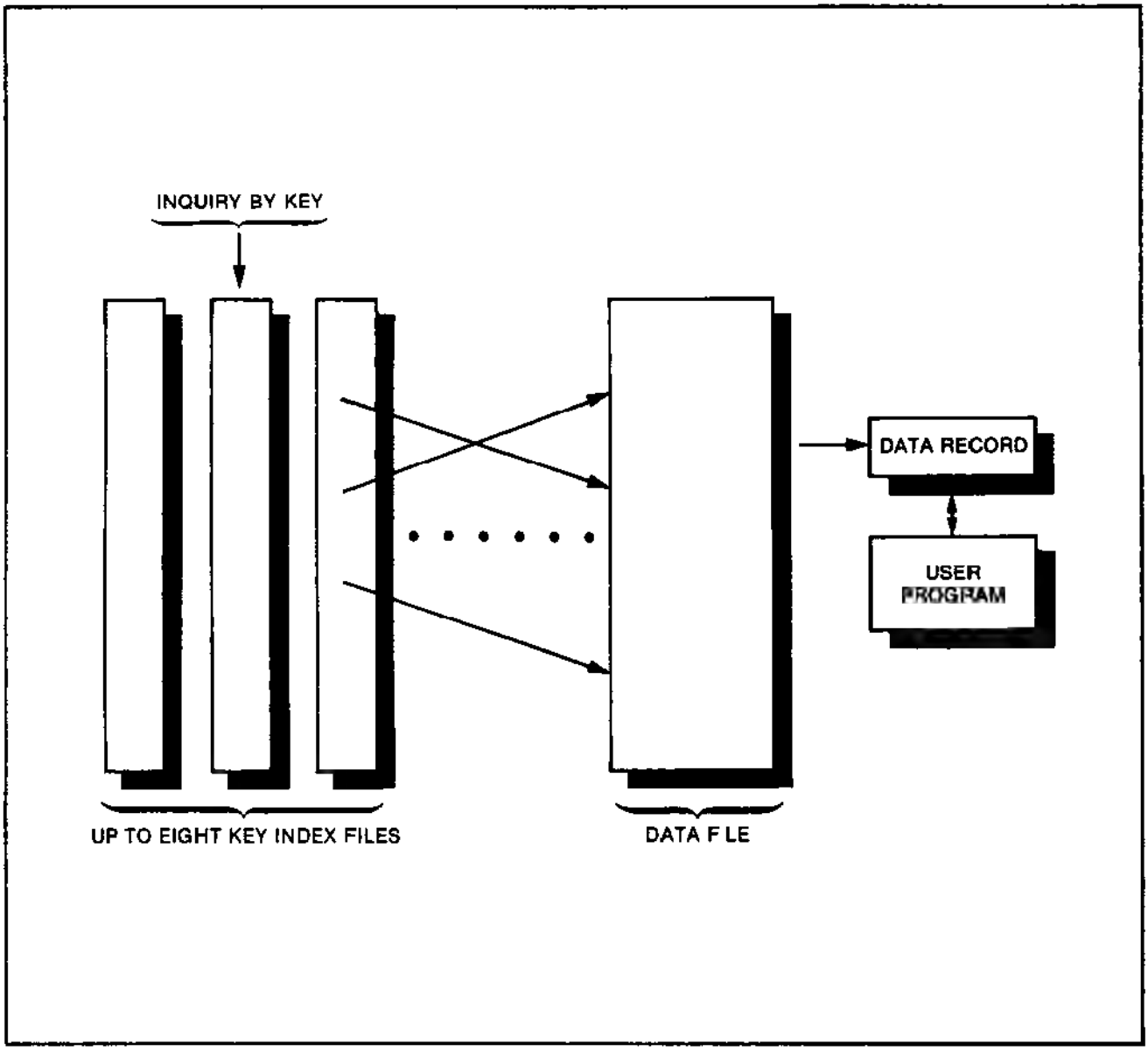


Figure 8-8. INFINITY Multi-Index Data Base File

QUERY can be used by people having little knowledge of computers. Operation of QUERY is simplified by an interactive HELP command that provides assistance and instructions on all facets of the program.

Refer to Figure 8-9 for an illustration of the structure of QUERY. For detailed information, refer to the QUERY INTERACTIVE INQUIRY AND REPORT WRITER, Reference and Technical Manuals, listed in the Preface.

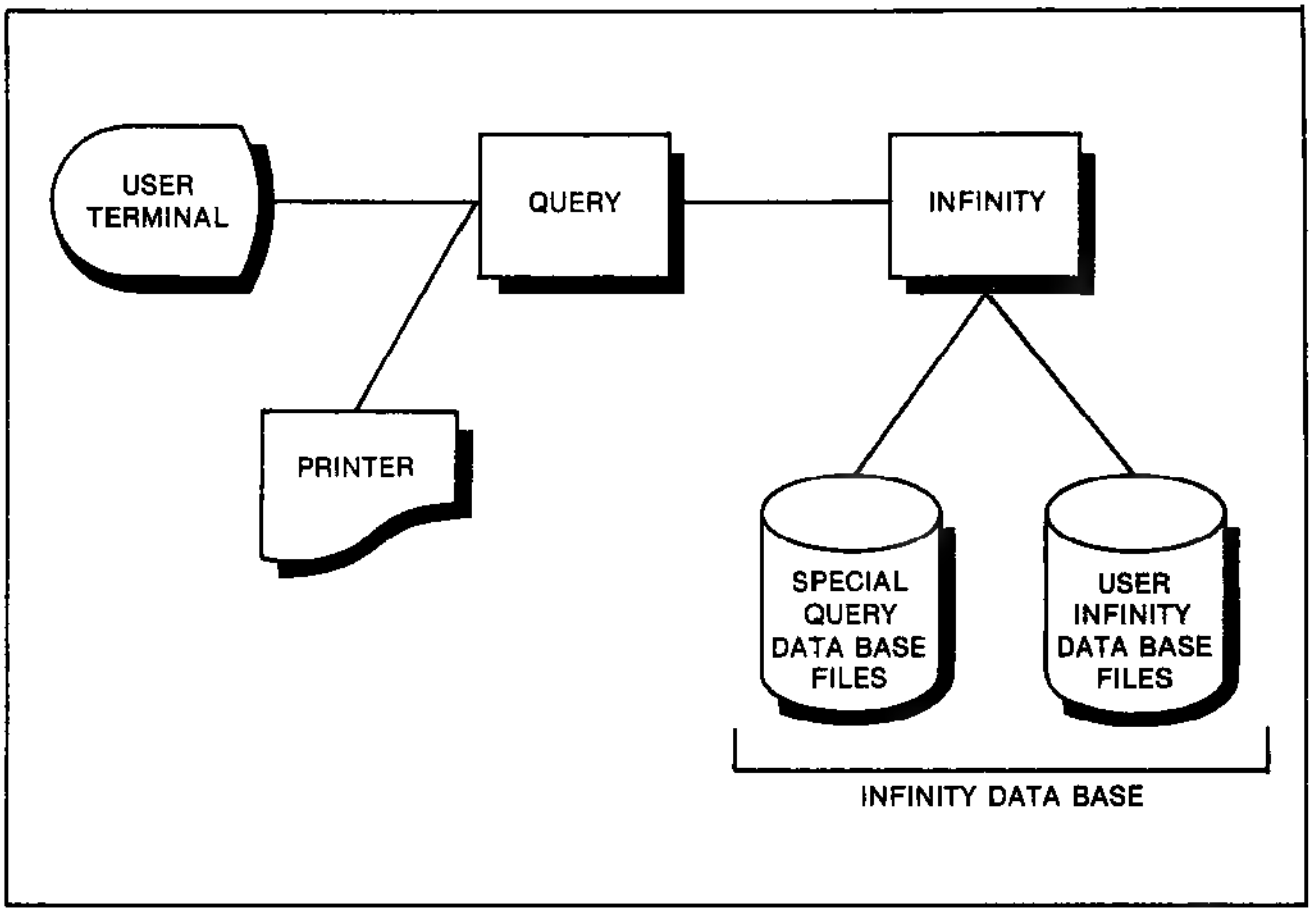


Figure 8-9. QUERY System Structure

8.3.4 TIME SHARING EXECUTIVE/TRANSACTION PROCESSOR (TSX)

The Time Sharing Executive/Transaction Processor (TSX) extends the capabilities of the MAX IV OS to support the interactive user in the following areas:

- o Time sharing and transaction-oriented processing.
- o Forms generation. Simplifies creation and maintenance of forms for CRT screens.
- o Extensive CRT support:
  - Interactive and buffered modes
  - Attributes (including color)
  - Function keys
  - Programmable status line
  - Line graphics
  - Auxiliary port - slave printers
- o Terminal/printer spooling.
- o Virtual terminal support. User tasks are independent of terminal control codes.

- o Password security access. Additional levels of user-defined security are permitted.
- o Master terminal control. The Master terminal controls the operation of all users to permit or restrict system access.
- o Pre/Post Processing. User programs can execute automatically at log-on and log-off.
- o File Manager support.
- o Language independent interface. FORTRAN, COBOL, Assembler, Pascal, and so forth.

For detailed information, refer to the TSX TIME SHARING EXECUTIVE AND TRANSACTION PROCESSOR, Programmer's Reference Manual, listed in the Preface.

8.4 LANGUAGE SUPPORT

The languages supported under the MAX IV OS are:

- o MODCOMP FORTRAN 77
- o PASCAL
- o COBOL 74
- o CORAL 66

8.4.1 MODCOMP FORTRAN 77

The MODCOMP FORTRAN 77 software package includes the compiler (FTN16) and a library of run-time routines. The FTN16 compiler provides a high degree of compatibility with MODCOMP FORTRAN IV (FR5).

The FORTRAN 77 compiler can operate on either the MAX IV (on any CLASSIC II or CLASSIC 32/85) or MAX 32 (on a CLASSIC 32/85) real-time operating system. The FORTRAN 77 compiler produces high-quality object code and maximum compatibility with the FORTRAN 66 standard and the MODCOMP FORTRAN compilers.

For detailed information, refer to the MODCOMP FORTRAN 77, Programmer's and Language Reference Manuals, listed in the Preface.

8.4.2 PASCAL

PASCAL is a general purpose, high-level programming language designed to simplify the writing of programs requiring complex data structures.

In addition, PASCAL is well-suited to the use of structured programming techniques, such as simple control structures, top-down design, and step-wise refinement.

MODCOMP PASCAL provides additional facilities necessary for PASCAL's use in real-time applications. These facilities include:

- o Separate compilation.
- o Access to FORTRAN IV and assembly language subroutines.

In addition, a PASCAL Cross-Reference Program (PXREF) is supplied. This program reads PASCAL source from logical file SI and displays a cross-reference of the source on logical file LO. The cross-reference consists of an alphabetic list of the identifiers used in the source and a list of source line numbers on which each identifier occurs.

For detailed information, refer to the PASCAL Language Reference Manual and the PASCAL User's Guide, listed in the Preface.

8.4.3 COBOL 74

MODCOMP COBOL 74 consists of a compiler and run-time support. The compiler has language error diagnostics and user options. The run-time support interfaces to the MAX IV Sort/Merge System and to the MAX IV I/O system, including the File Manager Subsystem.

MODCOMP COBOL 74 supports the requirements of ANSI COBOL (X3.23-1974). The following modules are implemented at the highest level:

- Nucleus
- Table handling
- Relative I/O
- Sort/Merge
- Segmentation
- Library

The MAX IV Sort/Merge System (SMS) is a system processor that performs sort/merge operations. SMS can be implemented as:

- A standard system processor
- A batch processor with interfaces to user-written code at various exit points
- A callable library routine

For detailed information on COBOL 74, refer to the MAX IV COBOL 74, Programmer's and Language Reference Manual. For detailed information on SMS, refer to the MAX IV SORT/MERGE SYSTEM, Programmer's Reference Manual, listed in the Preface.

8.4.4 CORAL 66

CORAL 66 is a general purpose, high-level, block-structured programming language particularly suited for real-time applications and system software development. The compiler has been approved by the IECCA British Government Committee as compatible with the official definition of the language.

CORAL 66 includes:

- TABLE facility
- Bit-manipulation
- Data overlaying
- Floating numbers
- Recursive procedures

In addition, MODCOMP has implemented the following extensions:

- Byte arrays
- Shift operators
- Hexadecimal constants
- Multi-named COMMON
- FORTRAN interface

CORAL statements provide powerful facilities to support the structured programming concepts of sequential statements, block structure, conditional branching, and iteration of loops. The principal types of statements are:

- Assignment statements to set values in variables.
- Procedure call statements.

A library of I/O routines is provided and the FORTRAN IV library may also be called. This provides not only I/O flexibility, but also the full range of mathematical routines available to FORTRAN IV.

For detailed information, refer to the MODCOMP CORAL, Language Reference Manual, listed in the Preface.

CHAPTER 9
MAX IV TECHNICAL PUBLICATIONS STRUCTURE

This chapter explains the set of technical publications associated with the MAX IV OS. Each manual type is described. A list of all manuals is presented, complete with a synopsis of the contents of each. In addition, Figure 9-1 illustrates the relationships among the manuals in the set.

The following types of manuals support the MAX IV Operating System:

- Concepts and Characteristics (CCM)
- System Guide (SGM)
- Programmer's Reference Manual (PRM)
- System Design Manual (SDM)
- File Management Manual (FMM)
- Library Manual (LBM)
- Language Reference Manual (LRM)
- Communications Reference Manual (CRM)
- Data Management Manual (DMM)
- Pocket Guides

9.1 CONCEPTS AND CHARACTERISTICS (CCM) MANUAL

The Concepts and Characteristics Manual is an overview document that summarizes the design and functionality of a system. In addition, this manual is a prerequisite to an understanding of the entire publications set for the system.

The objectives of this manual type are:

- To present concisely the principal concepts of the system as an aid to a basic understanding of the overall system functions.

- To orient the user with respect to system structure and terminology and to prepare the user for effective use of detailed manuals.

The audience of this manual type includes:

- Decision makers
- Customer application and system programmers
- System analysts
- Data processing management
- Sales and system personnel

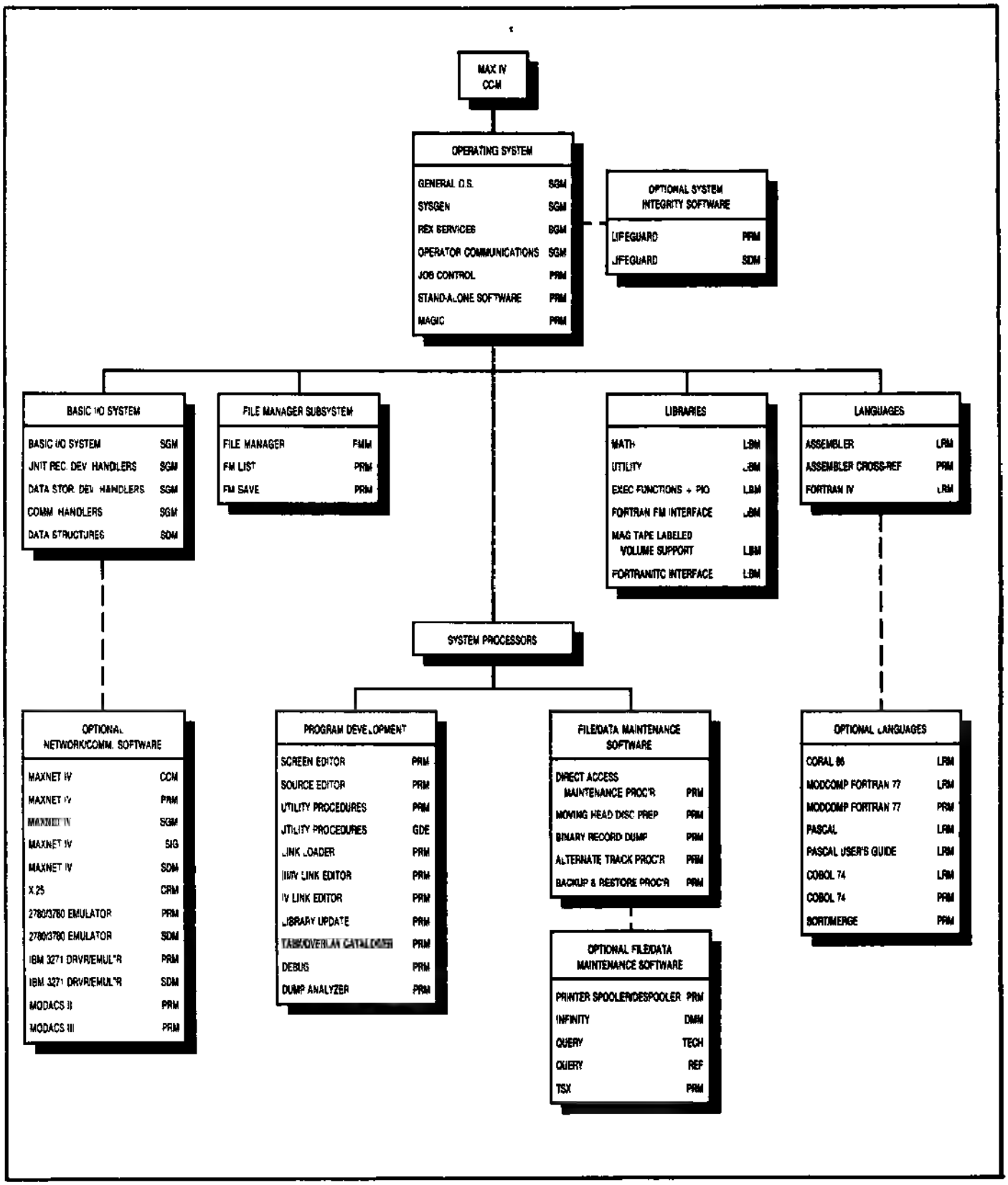


Figure 9-1. MAX IV Technical Publications Set

9.2 SYSTEM GUIDE (SGM)

The System Guide provides the operational description of the system software. The objective of this manual type is to provide reference information needed to operate the computer system under control of the operating system.

The audience of this manual type includes:

- Supervisors of computer operation
- Operators
- Programmers

9.3 PROGRAMMER'S REFERENCE MANUAL (PRM)

Two types of Programmer's Reference Manuals exist in the MAX IV publications set:

- PRMs for languages
- PRMs for system processors

The Programmer's Reference Manual for languages provides programming information about high-level or assembly languages. This manual type allows the programmer to utilize the system for user-specific applications. The objective of this manual type is to provide the programmer with information needed to prepare and update programs in a particular language for execution on a MODCOMP system. The audience of this manual type consists of programmers and system analysts.

The Programmer's Reference Manual for system processors provides information on the use of the system processor. The primary subject matter of this manual type is the directives within the system processor. This manual allows the programmer to utilize the system processor for user-specific applications. The audience of this manual type consists of programmers and system analysts.

9.4 SYSTEM DESIGN MANUAL (SDM)

The System Design Manual documents the logical design of a software system, providing an in-depth definition of the system structure to facilitate system maintenance, enhancements, and/or modification. The organization of the manual is based on data structures with an emphasis on dynamic relationships between components.

The objectives of this manual type are:

- To aid support personnel in troubleshooting.
- To document the detailed structure and flow of a system as a basis for modification (improvement, extension, or customizations).

The audience of this manual type consists of:

- Software developers
- Systems and support personnel
- Sustaining engineers
- Field personnel
- Customer systems analysts

9.5 FILE MANAGEMENT MANUAL (FMM)

The File Management Manual documents the file handling facilities, such as the MAX IV File Manager Subsystem, provided with an operating system.

The objectives of this manual type are:

- To describe the organization, use, and modification of files.
- To define procedures for the design, planning, and maintenance of file systems.
- To define the file-control and record-control capabilities of a system.
- To explain how and when to reorganize and restructure files.

The audience of this manual type consists of systems analysts and programmers.

9.6 LIBRARY MANUAL (LBM)

The Library Manual documents the subroutines contained in the various libraries associated with the MAX IV OS. Generally, each subroutine is described in terms of:

- Function
- Syntax
- Arguments
- Examples

The audience of this manual type consists of system programmers.

9.7 LANGUAGE REFERENCE MANUAL (LRM)

Language Reference Manuals document compilers and assemblers. These manuals define all language elements and their use (semantics and syntax). Language Reference manuals contain the following information:

- A brief overview of the language and its applications.
- Syntax of the language, described in detail.
- The coding form, described and shown with instructions for writing and programming statements.
- Illustrations of sample coded statements on the form to support the text.

The objectives of this manual type are:

- To provide a complete definition of all language elements in a specified version of the subject language, plus the rules for combining them to form program statements.
- To assist programmers in the use of the subject language.

The audience of this manual type consists of system analysts and programmers.

9.8 COMMUNICATIONS REFERENCE MANUAL (CRM)

The Communications Reference Manual contains all programmer-oriented information relative to a communications-oriented product such as emulators and drivers. The objectives of this manual type are:

- To provide descriptive information so that the programmer/analyst can design and implement the communication system.
- To enhance user applications.

The audience of this manual type consists of systems analysts and programmers.

9.9 DATA MANAGEMENT MANUAL (DMM)

The Data Management Manual presents the concepts, conventions, and formats of data types and files within data base management systems. The objectives of this manual type are:

- To describe data management facilities and how they control the allocation, organization, and retrieval of information in files.
- To describe formats of data types and files.
- To help in the most efficient utilization of file data.

The audience of this manual type consists of data base analysts and programmers.

9.10 POCKET GUIDES

A Pocket Guide is an abbreviated programmer's reference manual for a system processor. The objective of this manual type is to provide a quick reference tool. Therefore, pocket guides contain the following for each directive within the system processor:

- A very brief description of each directive.
- The syntax of the directive.
- Brief explanations of each parameter within the directive.

The audience of pocket guides consists of system programmers with experience in the use of the system processor.

9.11 MAX IV PUBLICATIONS SET

Table 9-1 lists the full names of the manuals in the MAX IV OS technical publications set and a synopsis of the table of contents of each manual. Refer to the latest Publications Catalog for manual kit numbers, manual order numbers, and ordering information.

The standard format of system processor manuals contains the following items:

- Functional overview/introduction to the processor.
- A summary of directives by category.
- Detailed descriptions of each directive including function, syntax, parameters, and examples.
- A guide to the use of the processor.

Table 9-1
MAX IV Technical Publications

TITLE	CONTENTS
Operating System Manuals:	
MAX IV General Operating System, SGM	In-depth overview; hardware requirements; memory management facilities; interrupt and trap processing; Terminal Monitor Program (TMP); Event Logging; Intertask Communication Service; Module Loader; standard names for device, controllers, and logical files.
MAX IV SYSGEN, SGM	An overview of the system generation process; description and syntax of each SYSGEN statement; information on transient partition definition; error messages; alphabetical reference guide; example SYSGEN files.
MAX IV Executive (REX) Services, SGM	Introduction to the types of REX services and calling conventions; detailed reference information on each service including service performed, calling parameters, programming considerations, condition codes returned, and examples; theory of operation; entry and exit macros and subroutines; numeric reference guide.
MAX IV Operator Communications, SGM	Overview of the Operator Communication (OC) Task; a summary of OC Directives; detailed reference information on each directive including description, syntax, parameters, and examples; information on writing custom directives; alphabetical reference guide.
MAX IV/MAX 32 Nonresident Job Control and Batch Facilities, PRM	An overview of Job Control including the types of Job Control directives; summary of directives by type; detailed reference information on each directive including purpose, syntax, and examples; a guide to the use of Job Control; detailed information on File Manager File Descriptors; Job Control and File Manager error messages.

TITLE	CONTENTS
MAX IV Data Structures, SDM	An overview of the types of MAX IV data structures; presentations of all data structures grouped according to the following categories: - Dedicated memory cells - Dedicated register blocks - General system data structures - Basic I/O System - Task-Related Data Structures - Specific System Task Data Structures In addition, user hooks and standard naming conventions for external labels and data structure elements are included in the manual.
MAX IV MAGIC, PRM	A guide for the MAX IV user on how to build a multi-batch system that best fits the user's needs. The manual describes, in detail, all the features of MAGIC, such as Electronic Mail and Text Formatter, plus it includes procedures and example programs for configuring the user's personal multi-batch system.
Stand-Alone Support Software:	
MODCOMP Stand-Alone Support Software, PRM	Reference information on all Stand-Alone Support Software such as the Disc/Tape Copy Program, Stand-Alone Linking Loader, and the Alternate Track Initializer. Information generally covers the configuration, and console switch settings.
Basic Input/Output System Manuals:	
MAX IV Basic I/O System, SGM	An introduction to the Basic I/O System including discussions of logical files, devices, naming conventions, and the User File Table; information on writing handlers; symbiont tasks.

TITLE	CONTENTS
MAX IV Unit Record Device Handlers, SGM	Introduction to Unit Record Device handlers; in-depth reference information on the following handlers: - Half-Duplex Teletypewriter/CRT - Line Printer - Raster Printer/Plotter - Incremental X-Y Plotter - Imaginary Printer Spooling Symbiont - Input Spooling Symbiont
MAX IV Data Storage Device Handlers, SGM	Introduction to Data Storage Device handlers; in-depth reference information on the following handlers: - Magnetic Tape Device - CLASSIC Disc Device - Fixed and Moving Head Disc Device - Large Capacity Moving Head Disc - Bulk Memory Device
MAX IV Communications Handlers, SGM	Introduction to Communications Handlers; in-depth reference information on the following handlers: - Asynchronous - Bisync - SDLC/HDLC - Standard Link Frame Level - Computer Link/Data Terminal - Local Process I/O
File Manager Manuals:	
MAX IV File Manager, FMM	Functional introduction to the File Manager; an overview and detailed information on File Manager services; file descriptors; error messages; overviews of all File Manager Utilities; a guide to the use of File Manager; SYSGEN statements; optional features of the subsystem including the Security, the Catalog, and the Audit packages.

TITLE	CONTENTS
MAX IV/MAX 32 File Manager File Listing (FMLIST) Utility, PRM	An overview of the Listing Utility; summary of directives by functional category; in-depth discussions of each directive including a functional description, syntax, and examples; a guide to the use of FMLIST; error messages and installation information.
MAX IV/MAX 32 File Manager Save/ Restore (FMSAVE) Utility, PRM	The contents of this manual parallel the contents of the FMLIST manual.
System Processor Manuals:	
Program Development	
MAX IV Source Editor, PRM	This manual contains an introduction to the Source Editor, a summary of the directives in the processor, a guide to the use of the processor, and detailed descriptions of each directive. The guide contains such topics as modes of operation, file copying and listing, and source updating. The detailed descriptions of each directive contains a functional description, the syntax, and examples. Error messages are also documented.
MAX IV MODCOMP Screen Editor, PRM	This manual follows the standard format outlined above.
MAX IV Utility Procedures, PRM	This manual follows the standard format outlined above.
MAX IV Utility Procedures, Pocket Reference Guides	These two pocket guides provide a quick reference tool for the experienced user of the Utility Procedures. Each directive is documented in terms of syntax and parameters only. The user is sent to the Programmer's Reference Manual for examples and detailed information.

TITLE	CONTENTS
MAX IV Link Loader, PRM	This manual contains detailed information on the logical flow and search techniques of the Link Loader. In addition, use of the Link Loader through Job Control is also provided.
MAX III/IV Link Editor, PRM	This manual contains information on the function and use of the processor. Limitations, error messages, and use of the Link Editor through a Job Control procedure are also provided. Directive discussions contain the standard format information.
MAX IV Link Editor, PRM	Under the guide to the use of this processor, items such as segmentation, common blocks, and extended memory are addressed. Directives and error messages are also documented.
MAX IV Library Update, PRM	This manual follows the standard format outlined above. Topics covered in the user guide chapter include operating conventions, directory libraries, sequential libraries, listing formats, and examples of Job Control procedures.
MAX IV Task/Overlay Cataloger, PRM	This manual follows the standard format outlined above. Topics covered in the user guide chapter include types of load modules, cataloging an overlay, cataloging a task, and Job Control procedures.
MAX IV Debug, PRM	This manual contains discussions on such topics as breakpointing, snapshotting, tracing, and stepping. Each directive and error message used by the processor is documented.
MAX IV Dump Analyzer, PRM	This manual follows the standard format outlined above. Topics covered in the user guide chapter include interfaces among the components of the processor, files referenced, performance characteristics, and installation. Appendixes include such information as dump options, address specifications, and error messages.

TITLE	CONTENTS
File/Data Maintenance	
MAX IV Direct Access Maintenance Processor, PRM	This manual includes such items as Maintenance Processor, PRM an overview, special data formats, user options, and installation procedures. In addition, directives are documented according to the standard discussions of function, syntax, parameters, and examples.
MAX IV Moving Head Disc Preparation Program, PRM	This manual follows the standard Preparation Program, PRM format outlined above. Topics covered in the user guide chapter include logical files and modes of executing the program.
MAX IV Binary Record Dump, PRM	This manual includes such topics as use, listing format, user options, error messages, and generation procedures.
MAX IV Backup and Restore Processor, PRM	This manual includes such topics as copy processing, run procedures, tape description and format, user interrupt capabilities, and error messages.
MAX IV Alternate Track Processor, PRM	This manual follows the standard format outlined above. Topics covered in the user guide chapter include mixed media drives and examples of use. Appendixes include such items as installation procedures and map structures.
Library Manuals:	
MAX IV Math Library, LBM	The general characteristics and use of the library is described; each subroutine is described in terms of function, syntax, arguments, and examples.
MAX IV Utility Library, LBM	The format of this manual parallels that of the Math Library manual.

TITLE	CONTENTS
MAX IV Executive Functions and Process I/O, LBM	The format of this manual parallels that of the Math Library manual.
MAX IV FORTRAN Interface to File Manager Library, LRM	An introduction to the library; detailed reference information on Service, File Descriptor Langauge, and Error Processing routines including the purpose, the calling sequence, programming considerations, and examples.
MAX IV Magnetic Tape Labeled Volume Support, LBM	An overview of the library is presented; the tape processing block is discussed; each subroutine is described.
MAX IV FORTRAN Interface to ITC Library, LBM	The format of this manual parallels that of the Math Library manual.
Language Manuals:	
MODCOMP Assemblers, LRM	An introduction covering compatibility, features, and listing format; types of addressing; directives presented in the following categories: - Definition - Data definition - Assembly control - Macros - Source input control
MAX IV Assembler Cross-Reference Program, PRM	An introduction covering symbols and system configuration; basic requirements, limitations, and procedures in the program; output from the program.
MAX IV FORTRAN IV, LRM	A general description of the compiler; a discussion of the coding and listing formats; conventions for data names and types; the syntax for statements in the following categories: - Declaration statements - Expressions - Assignment statements - Control statements - Input/output

TITLE	CONTENTS
MAX IV FORTRAN IV, LRM (continued)	Programs and subprograms; inline symbolic coding; guide to the use of FORTRAN IV.
Optional Support Software Manuals:	
Redundant Software System	
LIFEGUARD, PRM	Introduction; system operator directives; processing within the LIFEGUARD Executive Program; supported devices and SYSGEN requirements of the Dual Input/Output System; error messages.
LIFEGUARD, SDM	Purpose and system architecture; Stall Detector Program; overlays, routines, and data structures of the LIFEGUARD Executive Program; Dual Input/Output Symbiont routines; error processing; and communication routines.
Communications	
MODCOMP X.25, CRM	Brief introduction to the history and concepts of X.25; user program interface; OC directives; event reporting; X.25 SYSGEN statements; Call Control SYSGEN statements; installing the system; performance and limitations.
MAX IV 2780/3780 Emulator, PRM	Introduction and background; directives and parameters according to the following categories: - File/device control - Data transfer - Emulator control Generation and installation is also discussed.
MAX IV 2780/3780 Emulator, SDM	System characteristics; theory of operation; binary synchronous communication protocol; data structures.

TITLE	CONTENTS
MAX IV 3271 Driver/Emulator, PRM	Overview in terms of operating environment and features; the Driver Emulator; the Control Unit Emulator; programming considerations; SYSGEN procedures.
MAX IV 3271 Driver/Emulator, SDM	Bisync procedures; tracing emulator operations; SYSGEN information; data structures; conversion and protocol tables; naming conventions.
MODACS II, PRM	General description of functions supported by MODACS II and its support libraries. Includes data structures used and guide to building system using MODACS II. Description of subprogram purpose and calling sequence. System generation and hard-ware and firmware technical information included.
MODACS III Process I/O Subsystem, PRM	General description; handler functions; subprograms discussed in terms of purpose and calling sequence; process event monitoring; generation and installation.
MAXNET III/IV, CCM	A detailed introduction to MAXNET including discussion of a typical MAXNET environment, throughput and overhead, protocol, disc organization, and software/hardware considerations.
MAXNET IV, XRM	Detailed reference information on the use of REX I/O requests, FORTRAN interface routines, coredevices, and operator communication directives.
MAXNET IV, SGM	Detailed information on the syntax of the following types of directives: - Task Control - Remote File Control - Linkfile - Remote Batch - Link Analysis - Polling Control Information on error messages and down-filling of satellites is provided.

TITLE	CONTENTS
MAXNET IV, SIG	An overview of MAXNET IV installation, detailed information on SYSGEN statements, and installation examples for a variety of configurations is provided. In addition, discussions are included on the User Initialization Hook, Remote Batch installation, error analysis, and link tracing.
MAXNET IV, SDM	Detailed information on data structures is provided in two categories, general system and input/output. In addition, the theory of operation, standard naming conventions, MAXNET protocol, and debugging are included.
Optional File/Data Maintenance:	
MAX IV Printer Spooler/Despooler Subsystem, PRM	This manual follows the standard format outlined above. Topics covered in the user guide chapter include discussions of the Spooler, Queue Manager, and Despooler Symbionts.
INFINITY, Data Base Management System, DMM	This manual is structured according to the utility programs within the INFINITY system. The introduction includes such items as distributed data base networks, data base file design, and multi-user interaction. The following utilities are documented: - CREATE - ENTER - INQUIRE - DBCOPY - DBARCH - DBINIT - JOURNAL Installation of INFINITY is also discussed.
QUERY, Interactive Inquiry and Report Writer, XRM	This manual covers such topics as the operation of QUERY, commands, examples, and error messages.

TITLE	CONTENTS
QUERY, Interactive Inquiry and Report Writer, DMM	This manual covers such topics as descriptions and definitions of the Directory, Dictionary, Relationship, and Help file layouts. QUERY installation is also covered.
TSX, Time Sharing Executive/Transaction Processor, PRM	This manual covers such topics as general characteristics of the system, using the system, and system generation of TSX. In addition, the following is also discussed: - Preparing TSX files - System operator functions - Custom elements - The TSX Scheduler - Logging TSX usage
Languages:	
MODCOMP FORTRAN 77, LRM	FORTRAN concepts; data types; expressions; assignment statements; specification and data statements; control statements; I/O concepts; explicit and list-directed formatting; main programs and subprograms.
MODCOMP FORTRAN 77, PRM	Overview containing compiler system use, I/O, link-editing and cataloging a program, storage allocation and data formats, runtime I/O, mixed-language programming and error handling.
PASCAL, LRM	Language issues; data types; data declarations; expressions; statements; subprograms; I/O; storage manipulation.
PASCAL User's Guide, LRM	A guide to the use of PASCAL; information specific to MODCOMP PASCAL; programming practices; Cross-reference program.

TITLE	CONTENTS
MAX IV COBOL 74, LRM	Introduction to the parts of the COBOL program; file, record, and data concepts; the structure of the language; the IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE Division statements; the COBOL library; DEBUG; Sort/Merge; segmentation.
MAX IV COBOL 74, PRM	Introduction; system resources; compilation workflow; program execution; implementation details and restrictions; error messages.
MAX IV Sort/Merge System, PRM	A functional overview of the processor; descriptions of the directives including purpose, syntax, and examples; a guide to the use of Sort/Merge.
MODCOMP CORAL, LRM	An overview of the language; data types; expressions in the language; statements; I/O; library interfaces; operational aspects.

APPENDIX A
ACRONYMS

ACRONYM	DEFINITION
ANSI	American National Standard Institute
BIOS	Basic Input/Output System
BRD	Binary Record Dump Processor
BRP	Backup and Restore Processor
CAN	Compressed Alpha-Numeric
CCM	Concepts and Characteristics Manual
CRM	Communications Reference Manual
CU	Control Unit (3271)
DAMP	Direct Access Maintenance Processor
DIOS	Dual Input/Output Symbiont
DLR	Down Line Remote Task
DMP	Direct Memory Processor
EPA	Error Processing Address
FAT	File Assignment Table
FDL	File Descriptor List
FIFM	FORTRAN Interface to File Manager
FM	File Manager
FMLIST	File Manager File Listing Utility
FMM	File Management Manual
FMSAVE	File Manager Save/Restore Utility
FR5	FORTRAN IV
GFAT	Global File Assignment Table
ITC	Inter-Task Communications Service
JC	Job Control
LBM	Library Manual
LIB	Library Update
LPR	Load Program Records
LRM	Language Reference Manual
M5A	Macro Assembler
ML	Module Loader
MODACS	Modular Data Acquisition and Control Subsystem
MOR	Module Object Records
MSED	MODCOMP Screen Editor
OC	Operator Communications Task
OS	Operating System
PIO	Process Input/Output
PR	Program Address Register
PREP	Moving Head Disc Preparation Processor
PRM	Programmer's Reference Manual
PS	Program Status Register
PXREF	PASCAL Cross-Reference Program
REL	Remote Loader
REX	Request Executive Service
RJE	Remote Job Entry
ROL	Roller Task
S	Output Spooler Task
SAL	Stand-Alone Loader

SDM	System Design Manual
SED	Source Editor
SGM	System Guide
SMS	MAX IV Sort/Merge
SYSGEN	System Generation
TCB	Task Control Block
TCU	Transmission Control Unit (3271)
TMP	Terminal Monitor Program
TOC	Task/Overlay Cataloger
TSX	Time Sharing Executive/Transaction Processor
VC	Virtual Circuit
X	Exceptional Condition Task
XREF	Assembler Cross-Reference Program

INDEX

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 3624 FT. LAUDERDALE, FL 33309

POSTAGE WILL BE PAID BY ADDRESSEE

MODULAR COMPUTER SYSTEMS
1650 W. McNAB ROAD
P.O. BOX 6099
FT. LAUDERDALE, FLORIDA 33310

Attention: TECHNICAL PUBLICATIONS, M.S. #85

Fold

MODCOMP®

Please comment on the publication's completeness, accuracy, and readability. We also appreciate any general suggestions you may have to improve this publication.

If you found errors in this publication, please specify the page number or include a copy of the page with your remarks.

Your comments will be promptly investigated and appropriate action will be taken. If you require a written answer, please check the box and include your address below.

☐

Comments: _____

Manual Title _____

Manual Order Number _____ Issue Date _____

Name _____ Position _____

Company _____

Address _____

_____ Telephone ( ) _____

MODCOMP

Corporate Headquarters:
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel: (305) 974-1380, TWX: 310-372-7837

European Headquarters:
MODULAR COMPUTER SERVICES, Inc., The Business Centre, Molly Millars Lane, Wokingham, Berkshire, RG11 2JQ, UK, Tel: 0734-786808, TLX: 851849149

Latin American Sales Headquarters:
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel: (305) 975-6562, TLX: 3727852

Canadian Headquarters:
MODCOMP Canada, Ltd., 400 Matheson Blvd. East, Unit 24, Mississauga, Ontario, Canada L4Z 1N8, Tel: (416) 890-0666, TELEX: 06-961279

SALES & SERVICE LOCATIONS THROUGHOUT THE WORLD

Printed in U.S.A.